Testing Geospatial R Packages on Implementations of the R Language and Platforms

Thesis submitted in partial fulfillment of the requirements

for the Degree of Master of Science in Geospatial Technologies

February 2020

Ismail Sunni

imajimatika@gmail.com

https://github.com/ismailsunni

Supervised by:

Edzer Pebesma

Institute for Geoinformatics

University of Muenster

Co-supervised by:

Daniel Nüst

Institute for Geoinformatics

University of Muenster

Co-supervised by:

Roberto Henriques

Information Management School

Universidade Nova de Lisboa

Declaration of Academic Integrity

I hereby confirm that this thesis on *Testing Geospatial R Packages on Implementations of the R Language and Platforms* is solely my own work and that I have used no sources or aids other than the ones stated.

All passages in my thesis for which other sources, including electronic media, have been used, be it direct quotes or content references, have been acknowledged as such and the sources cited.

February 24, 2020

_____

I agree to have my thesis checked in order to rule out potential similarities with other works and to have my thesis stored in a database for this purpose.

February 24, 2020

_____

Acknowledgements

I would like to say thanks to my God, Allah SWT, without His help and guidance, I will not be able to finish my thesis and my master study. I want to thank my supervisor, Edzer Pebesma, for giving early ideas and feedbacks. I want to thank Daniel Nuest as my co-supervisor, who is constantly supporting me whenever I need and encouraging me to do proper research. Lastly, Roberto Henrique for his valuable feedback on the thesis writing. Thanks also for the whole R and open source community for providing great software and tool and answer for my questions.

I wish to acknowledge the support from my family in Indonesia and especially my wife, Dian Rokhmawati, who is always there for me. I love you. And also my friends in Erasmus Mundus Geotech who is like a family for me.

Lastly, I want to say many thanks for the European Commission for granting me a scholarship which enables me to pursue this master's degree and work on this thesis. The scholarship also opens a lot of new doors and experiences for me.

Table of Contents

List of Tables

List of Figures

Abstract

R is a programming language for statistical computation and graphics. Besides the commonly used, GNU R, there are other alternative R implementations that claim to have advantages compared to the GNU R. Unfortunately, it is not clear how will geospatial R packages behave on these implementations since these packages often rely on system libraries which installed at the system level. System libraries also depend on the platform where the R is running. To find this information, this research aims to explore the compatibility of geospatial R packages on different R implementation and platform. This research also aims to see which R implementation and platform has the best performance.

To make the exploration easier, container technology is used to install system dependencies and R implementations. All system dependencies from `sysreqsdb` are installed for geospatial R packages. From this exploration, it is found that not all R implementations are compatible with geospatial packages. Problems found can be grouped into three categories: System Dependencies, Unsupported Implementation, and Running Time Error. GNU R and Microsoft R Open (MRO) are the only R implementations that compatibles with geospatial R packages.

A benchmarking R package called `altRnative` is created to run the benchmarking across the successful combination. The benchmark result shows that GNU R has a little bit better performance (1.2x) compared to MRO regardless of the platforms.

Reproducibility self-assessment (`https://osf.io/j97zp/`): 3, 3, 3, 3, 3 (input data, preprocessing, methods, computational environment, results)

Keywords: R implementation, benchmark, system dependencies, Docker, sysreqsdb

Chapter 1

Introduction

1.1 Motivation

R is a programming language for statistical computation and graphics (Hornik, 2018). R has more than 14.000 packages, which are available through the Comprehensive R Archive Network (CRAN) ("CRAN - Contributed Packages," 2019). There are over 2050 geospatial (spatial and spatial-temporal) packages to read, write, or analyze spatial data(Bivand, 2019). On CRAN, packages are published and tested to maintain their functionalities and ensure compatibility.

Currently, those packages are tested for different compilers and on different operating systems, but only for the main implementation of R by R Development Core Team and only for Debian, (Hornik, 2018). In CRAN, there are also result for R packages installation and checking for Debian GNU/Linux, Fedora, OS X, Solaris, and Windows (R Development Core Team, 2019).

There are other alternative implementation exist like pqR (Neal, 2019), Microsoft R Open (MRO) (Microsoft, 2019), Renjin (BeBetaDriven, 2019), fastR (Oracle, 2019b), and TIBCO Enterprise Runtime for R (TERR) (TIBCO, 2019). These alternatives implementations are created to make a faster R (Hadley Wickham, 2019). Besides the claimed faster speed, there are also some features of these implementations that can be useful for the user like better memory management (pqR), seamless integration to Java (Renjin), multithreading performance (MRO), and enterprise-grade analytics engine (TERR).

Unfortunately, it is not clear how will geospatial R packages behave on these implementations since these packages often rely on system libraries installed at the system level. For example, sf package depends on GDAL, GEOS, and PROJ (Pebesma et al., 2019). This means it's not sure whether they can work on different platforms and R implementations or not. The behavior refers to whether an R package can be installed, run, and its performance on these alternative R implementations. Besides the alternative R implementations, the platform where the R runs is also taken into consideration since the libraries are installed at the system level.

A container is computer software that is used to package and run an application and all of its dependencies (Docker, 2019). Since it is isolated from the external

system, users can set up anything that is needed and run even complex software environments on another computer easily. This ability enables the developer to create different computational environments to run specific software. In testing, it is very useful since you can have a well-defined configuration, especially if you need specific versions dependencies (Manuel Weiss, 2016).

1.2 Aim and Objective

The main goal of this research is to understand what the performance of geospatial R packages in non-base R implementations and on platforms currently not supported by CRAN is. The behavior that we want to understand is the installation and testing aspects of the R packages. For example, whether an R package can be installed or not and whether the packages' tests pass or not in the uncommon R implementations.

One step to achieve this goal is creating a Docker image for all possible combinations of R implementations and platforms. The next step is testing the compatibilities of packages from spatial and spatial-temporal views on the containers from the docker image created previously. Lastly, based on the result of the previous testing, the performance of a successful combination is benchmarked in terms of speed for a typical geospatial workflow.

1.3 Research Question

1. What are the system dependencies for the R packages for each different platform? Is the information provided by the sysreqs database complete for geospatial packages?
2. Which combinations of the platform (operating system distribution), R implementation, system libraries, and geospatial R packages that can be successfully installed?
3. What are the causes of unsuccessful installations?
4. For the compilable combination, how is the comparison between them in terms of performance?

Chapter 2

Background

In this chapter, various R implementations, Linux distributions, container technology, and benchmarking in R is explained.

2.1 R Implementations

In this part, various known alternative R implementations are introduced in brief.

2.1.1 GNU R

The main implementation of R programing language is developed by the R Core Team and belongs to the GNU project. The two latest release minor releases are 3.5.x and 3.6.x (Hornik, 2018). To avoid confusion, in this thesis the main implementation is called by GNU R.

2.1.2 Microsoft R Open

Microsoft Open R (MRO) is created by Microsoft Corporation which has additional capabilities for better performance, reproducibility, and platform support. The latest release (MRO 3.5.3) is based on GNU R 3.5.3. It has compatibility with all packages that compatible with GNU R 3.5.3(Microsoft, 2019). There are some Docker images available for example by (Nüst, February 26, 2016/2019) based on Ubuntu 18.04 and (Lisic, August 23, 2017/2017) based on CentOS 7.

2.1.3 Renjin

Renjin is a JVM-based interpreter for the GNU R. It is developed by BeDataDriven. Its main features are seamless integration between Java and R, big data processing, better performance on some operations, easiness to deploy in cloud infrastructure (BeBetaDriven, 2019). There is a list of R packages in (BeDataDriven, 2019) that show its compatibility and why it does not compatible. It shows that many geospatial R packages are not compatible with Renjin. There is a Docker image available for Renjin (Nüst, 2019a).

2.1.4 FastR

FastR is R implementation by Oracle that aims at a high-performance (Oracle, 2019b). It is built on GraalVM, a universal virtual machine to run applications created by many programming languages (Oracle, 2019a). There is a Docker image available for FastR (Nüst, 2019b).

2.1.5 pqR

pqR is another R implementation created by Radford Neal that compatibles with GNU R 2.15.1. It claims to improve GNU R in terms of speeding up many functions and perform some numerical computation in parallel on a multi-core processor (M. Neal, 2019). There is a Docker image available for pqR (Nüst, 2019c).

2.1.6 TIBCO Enterprise Runtime for R (TERR)

TERR is an enterprise-grade analytic engine created by TIBCO that has full compatibility with GNU R. It can most common R packages without modification (TIBCO, 2019). There is not know Docker image for this R implementation.

2.2 Linux Distribution

From 10 major Linux distribution based on (DistroWatch, 2019), Debian, Fedora, and Arch Linux are chosen for this research. Debian and Fedora are chosen because they are being used on CRAN and have different package managers. Arch Linux is chosen because it is a Linux distribution in different spectrum compared to Debian and Fedora (DistroWatch, 2019). Ubuntu is not chosen because it is built on Debian and is widely used in the GIS & R community (cf. UbuntuGIS). Fedora represents the Fedora/RedHat/CentOS distributions (Daniel Miessler, 2019).

2.3 Container

A container is computer software that is used to package and run an application and all of its dependencies (Docker, 2019). Since it is isolated from the external system, users can set up anything that is needed and run even complex software environments on another computer easily. This ability enables the developer to create different computational environments to run specific software. In testing, it is very useful since you can have a well-defined configuration, especially if you need specific versions dependencies (Manuel Weiss, 2016). An R package called `stevedore` can be used to work with the container by using the Docker API (FitzJohn, 2019).

2.4 Benchmarking in R

There are already several benchmarking R package for R like using `microbenchmark` (Mersmann, Beleites, Hurling, Friedman, & Ulrich, 2019), `rbenchmark` (Kusnierczyk, 2012), `tictoc` (Izrailev, 2014), and `bench` (Gossmann, 2017). Those packages measure the duration of running an R script or expression. Unfortunately, those packages do not support running the benchmark for different R implementation and platform easily.

Chapter 3

Methodology

In this chapter, the methodology of the research is explained. For better understanding, the methodology is shown as in 3.1. From this diagram, there are nine blocks that are grouped into 3 main parts. The first part is **Docker Images** that covers *Various R implementations*, *Platform*, *Various R + Platform Docker Images*, *Spatial and SpatioTemporal R packages* and *Geospatial R Docker Image*. From this first part, the answer to the first and second research questions can be found. The second part is **Benchmarking** which covers *Geospatial Workflow* and *Benchmarking* block. And the last one is *Analysis* that contains *Failure Analysis* and *Benchmark Analysis*. This third will answer the third and fourth research questions. As shown in figure 3.1, there four numbers that represent the location where the answer to each research question. In the following sections, each part is explained in detail.

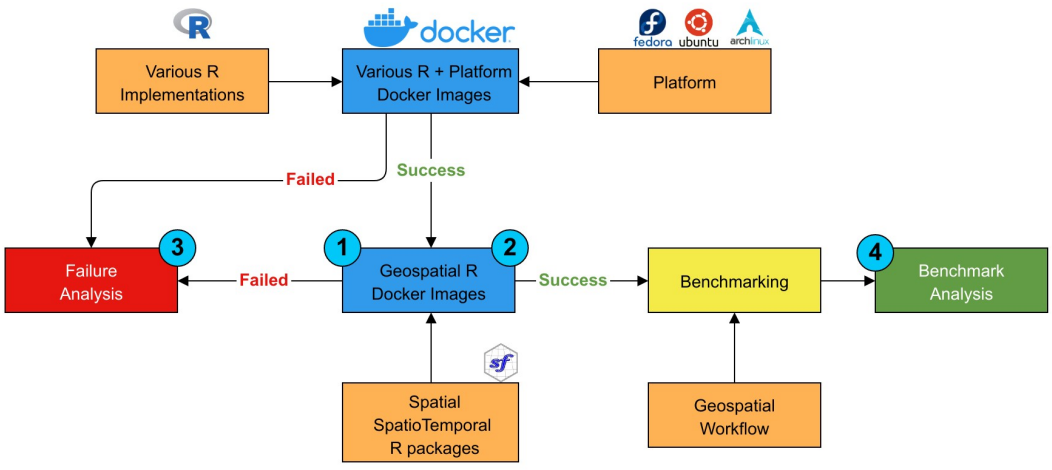

Figure 3.1: Methodology

3.1 Docker Images

The first step is creating Docker images for each combination of platform and R implementations without the geospatial R package. These Docker images (*Various R*

+ Platform Docker Images) are called Vanilla R Docker images for easier reference in this research. To create these Vanilla R Docker images, it needs to install the *Various R implementations* on the *Platform*. The research uses 6 R implementations (GNU R, MRO, Renjin, FastR, pqR, and TERR). For the platforms, Debian, Fedora, and Arch Linux. The reasons for choosing these platforms have been mentioned in 2.2. It is possible to use another platform with the same package manager if it's found that there is already a working Docker image or there is a problem with the current platform. These changes, for example, Ubuntu for Debian.

There are other popular platforms like Windows and macOS. Windows platform is not chosen because of the requirement of its Docker image. Windows docker image can be run only from Windows host machine[1]. MacOS is also not chosen because of its software license agreement[2].

To create the *Geospatial R Docker Image*, it needs to install geospatial R packages including their system dependencies. The list of dependencies is taken from ("R-hub/sysreqsdb," 2019) database. If there is an uninstalled system dependency or missing dependencies, they are noted and reported to the repository. *Spatial and SpatioTemporal R packages* are the packages under Spatial and SpatioTemporal view. They will be installed by using CTV packages (Zeileis, 2005) whenever possible. If it is not possible, installing one by one package will be done. All failed installations will be tried to fix or noted if it is not possible to fix it.

3.2 Benchmarking

Since there is no available tool to run a benchmark for R across different platforms, the tool needs to be created first. The idea is to create an R package that able to run R script inside a Docker container. It can be done by utilizing the `stevedore` R package. `stevedore` is an R package that can be used for Docker client manipulation(FitzJohn, 2019). For the benchmark part, `microbenchmark` can be used to handle the benchmarking process (Mersmann et al., 2019). `microbenchmark` also offers nice plotting functionality that can be used for easier analysis. `microbenchmark` only gives the raw value of elapsed time. For a better view of the performance comparison, an additional feature to get the ratio of the elapsed time is needed. This feature will be implemented also as well as the plot for the elapsed time ratio.

For the benchmark use case, the Spatial Data Science R (SDSR) book (Pebesma, 2019b) as the *Geospatial Workflow*. The idea is running all of the code inside the SDSR book as it represents a common geospatial workflow. Modification of the book can be done if there is an error or complicated prerequisite (e.g. downloading the additional package, downloading large data). The benchmark process is run on Dell XPS 9360 laptop with Intel® Core™ i7-8550U CPU @ 1.80GHz × 8 processor, 16 GB of RAM, and 512 GB of SSD. It runs Ubuntu 18.04. Any other unneeded application beside the benchmarking is killed so that they would not disturb the benchmarking

[1] `https://stackoverflow.com/questions/33190469/linux-machine-with-docker-deploy-windows-container/33190605#33190605`

[2] `https://www.apple.com/legal/sla/docs/macOSCatalina.pdf`

process.

3.3 Analysis

The last part is *Analysis* based on the result of the previous two parts. The first one is *Failure Analysis* which tries to analyze why a problem or an error occurs in the process of Docker image creation. This analysis covers the various R implementations installation, system dependencies installation, geospatial R package installation, and running the geospatial workflow. Some problems may be easily fixable, while others can only be fixed with a lot of work. An example problem that can only be fixed with a lot of work is the problem related to the R implementation itself or related to packages availability on a specific platform or R version. The general rule is if it's fixable within the research time frame, it will be tried to fix otherwise it will be marked as not fixable. All the problems and the fixes will be reported to their maintainer and discussed in the next chapter.

The second analysis is *Benchmark Analysis* which will cover the result of the benchmark. From the result, it can be analyzed what is the main factor for the performance in the case of the SDSR book. More detail analysis can be done also to see which chapter of the book that takes the most of the running time.

Chapter 4

Result and Analysis

In this chapter, the result of three parts from the methodology is shown. The first section explains the Docker images and the second section is about the benchmark result. The analysis part is included directly for those sections.

4.1 Docker Images

In this section, the exploration's result of each combination between R implementations and platforms for the R installation and geospatial packages installation is shown. The result is described per each R implementations. The overview of the result can be seen in figure 4.1 for Docker image without geospatial R package and 4.2 for Docker image with geospatial R package.

In those two tables, there are four kinds of results. The first one is **Yes** which means that the Docker image is successfully created and it works as expected for this theses. The second result is **No** which means that the Docker image is not possible to create properly. More detail of the failure is explained below. The third one is **Stop**. This result means the Docker image is not created because its geospatial R Docker image will not work properly based on the other Docker image result. And the last one is **?** which means, it may be possible to create the Docker image but is not created because of the time limit.

First, in order to gather all of the system dependencies required by geospatial R packages, R package `sysreqs` is used to access the `sysreqsdb` database. The result for each platform is obtained using this method. These results then used for creating geospatial R Docker images. The full result is available online[1].

All Docker images except the TERR one is published in Docker hub[2] while all Docker file is published in Github repository[3]. In the Github repository, there are both successful and unsuccessful Docker file. It also provides some notes about the detail of the error and some command to build, run, log in to the Docker container shell, and completeness check of the geospatial R packages (Sunni & Nüst, 2020b).

[1] `https://github.com/ismailsunni/dockeRs/tree/master/scripts/sysreqs`
[2] `https://hub.docker.com/u/ismailsunni`
[3] `https://github.com/ismailsunni/dockeRs/`

Table 4.1: Vanilla R Docker Images

	GNU R	MRO	Renjin	FastR	pqR	TERR
Debian	Yes	Yes	Yes	Yes	Yes	Yes
Fedora	Yes	Yes	Stop	Yes	Stop	Stop
Arch Linux	Yes	Yes	Stop	Stop	Stop	Stop

Table 4.2: Geospatial R Docker Images

	GNU R	MRO	Renjin	FastR	pqR	TERR
Debian	Yes	Yes	No	No	No	No
Fedora	Yes	Yes	No	No	No	No
Arch Linux	Yes	?	No	No	No	No

4.1.1 GNU R

Debian

There is a Docker image available for GNU R on Debian[4], so there is no need to work on this. This Docker image is created by Rocker Project[5]. This project also provides a GNU R Docker image with geospatial package[6]. Unfortunately, this Docker image does not contain all geospatial R package but only for a subset of it. The selected packages are the one which is slow or tricky to install and the one that has a general-purpose. The geospatial R Docker image for this thesis is created by extending the Rocker Project geospatial R Docker image (adding the missing package). In the end, both the Docker image for GNU R on Debian and its geospatial version can be created.

Fedora

By using the Docker image based on Fedora 32, the Docker image for GNU R on Fedora is created. Based on this Docker image, GNU R on Fedora with geospatial is created. There was a problem with installing system dependencies from `sysreqsdb`. The problem is `proj-epsg`, `proj-nad`, and `v8-314-devel` are no longer available. This problem is solved by removing the first two packages and changing the third on to `v8`. In the end, both the Docker image for GNU R on Fedora and its geospatial version can be created.

Arch Linux

Based on the `archlinux:20200106` Docker image, the GNU R Docker image on Arch Linux is created. From this Docker image, GNU R geospatial Docker image is created. There is a problem with installing system dependencies since not all of the

[4] `https://hub.docker.com/_/r-base`
[5] `https://www.rocker-project.org/`
[6] `https://hub.docker.com/r/rocker/geospatial`

system dependencies are available in the Arch Linux package database. These system dependencies can be installed from the Arch User Repository (AUR)[7]. To install a package from AUR on Docker image, it needs to create a user without a password. In the end, both the Docker image for GNU R on Fedora and its geospatial version can be created.

There is an interesting finding on Geospatial GNU R on the Arch Linux Docker image. There is a different behavior of R code in this Docker image compared to the one on Fedora and Debian. The difference is about the CRS comparison. It is caused by different versions of GDAL. The Fedora and Debian Docker image have GDAL 2.x while this Arch Linux has GDAL 3.x. The issue has been reported to `sf` issue tracker[8].

4.1.2 Microsoft R Open

Debian

There is a Docker image available fro MRO 3.5.3 on Ubuntu 18.04 (Nüst, February 26, 2016/2019). From this Docker image, MRO 3.5.3 with geospatial R packages is created. There are three problems found. The first one is `libav-tools` is no longer available on this Ubuntu version. It's changed with `ffmpeg`. Another issue is `gfortran` which is needed for some R packages is not found on the base image, so it should be installed first. The third one is the `polyclip` package can not be installed from within MRO[9]. It is solved by installing the `polyclip` package from the source. In the end, both the Docker image for MRO 3.5.3 on Ubuntu and its geospatial version can be created.

Fedora

There is no Docker image for MRO on Fedora. At first, it's created based on the latest version of Fedora (Fedora 32). It works properly until `sf` is tried to install. There is an error that mentioned `proj_api.h not found in standard or given locations`[10]. In the same Docker image, GNU R is installed to check whether the issue is MRO specific or the Docker image specific. It turns out that it is possible to install `sf` on that Docker image for GNU R. In the end, it is solved by using an older version of Fedora (Fedora 30). One possible explanation is that Fedora 32 has `proj` version 6.2 while Fedora 30 has `proj` version 5.2. MRO 3.5.3 is based on GNU R 3.5.3 which has better support for older `proj` version. In the end, both the Docker image for MRO 3.5.3 on Fedora 30 and its geospatial version can be created.

[7] `https://aur.archlinux.org/`
[8] `https://github.com/r-spatial/sf/issues/1238`
[9] `https://github.com/ismailsunni/dockeRs/issues/32`
[10] `https://github.com/ismailsunni/dockeRs/issues/39`

Arch Linux

The is no official support for Arch Linux from MRO but there is an MRO installer from AUR[11]. By using this package, a Docker image for MRO 3.5.3 on Arch Linux with tag 20200106 is created. Unfortunately, the Docker image for MRO 3.5.3 with the geospatial R package is not possible to create using this base image. The reason is that in MRO installed the pinned version of `sf` (based on R 3.5.3) on the snapshot. It expects to have GDAL 2.x which still has `pcs.csv`. This file is removed in GDAL 3.0 which is the current version in the latest Arch Linux.

There is a possibility to downgrade the installed GDAL version to match with the requirement of `sf` in MRO 3.5.3 snapshot. Unfortunately, since the discovery of the MRO installer on Arch Linux is very late, the geospatial Docker creation is not done due to the time limit. In the end, the Docker image for MRO 3.5.3 on Arch Linux is created but its geospatial version is not created due to time limit.

4.1.3 Renjin

Debian

Renjin Docker image for Debian is available (Nüst, 2019a). Unfortunately, Renjin doesn't support `sf` ("Renjin.Org | sf 0.7-4," 2020) because Renjin does not support `units` package ("Renjin.Org | units 0.6-3," 2020) which is needed by `sf`. `units` package is not installed even though `udunits2` is available because Renjin does not support packages that depend on external libraries, in this case, `udunits2`[12]. In the end, the Docker image for Renjin on Debian is created but its geospatial version can not be created.

Fedora

Since it is not possible to install geospatial packages on Renjin on Debian, the same situation applies for Renjin on Fedora. Because of this reason, the Docker image for Renjin on Fedora is not created and marked as *stop* in the final result.

Arch Linux

Since it is not possible to install geospatial packages on Renjin on Debian, the same situation applies for Renjin on Arch Linux. Because of this reason, the Docker image for Renjin on Arch Linux is not created and marked as *stop* in the final result.

[11] `https://aur.archlinux.org/packages/microsoft-r-open/`

[12] `https://stackoverflow.com/questions/58996151/rccp-h-not-found-in-renjin-when-compiling-package/59017512`

4.1.4 FastR

Debian

There is a Docker image for FastR 3.4.0 on Debian(Nüst, 2019b). But there is a problem with installing the `sf` package (one of the core geospatial packages). The problem is related to `proj` not found although it's already installed.[13] The next attempt is using the latest FastR release 3.6.0, by extending an example from.[14] Unfortunately, the same problem still exists. To check whether this problem is FastR specific or not, GNU R is installed for both Docker images and it's possible to install `sf` on it. This means the problem is specific to FastR. In parallel, a Docker image for FastR on Fedora is tried to create, but it is not possible. A more detail explanation is shown in the Fedora part below. In the end, it is possible to create a Docker image for FastR on Debian but it is not possible to create its geospatial version on Debian.

Fedora

Oracle Linux 7 has the same package manager as Fedora, so Oracle Linux 7 is also qualified as the Fedora-based Docker image in this research. Using the official GraalVM Docker image ("Oracle/graalvm-ce - Docker Hub," 2019) based on Oracle Linux 7 it is possible to install FastR 3.6.1. But geospatial packages like `sf` is not possible to install because the GDAL is too old on Oracle Linux 7. It is solved by building GDAL from source[15]. A similar problem (old package) also happens with `Pandoc` that is used for running the book knitting in SDSR. This is not mandatory though for geospatial workflow.

Unfortunately, another problem occurs when running the R script from the SDSR book. It is found out that `sf` in FastR can not read a geopackage file. The problem has been reported to FastR repository[`https://github.com/oracle/fastr/issues/128`], but there are no comments from the maintainer or other people. From these results, it is concluded that it is not possible to install geospatial packages properly on FastR. In the end, it is possible to create a Docker image for FastR on Fedora but it is not possible to create its geospatial version on Fedora.

Arch Linux

Since it is not possible to install geospatial packages on FastR on Debian and Fedora, the same situation applies for FastR on Arch Linux. Because of this reason, the Docker image for FastR on Arch Linux is not created and marked as *stop* in the final result.

[13] `https://github.com/oracle/fastr/issues/116`
[14] `https://github.com/graalvm/examples/issues/6`
[15] `https://gist.github.com/simondobner/f859b2db15ad65090c3c316d3c224f45`

4.1.5 pqR

Debian

There is already a pqR Docker image for Debian(Nüst, 2019c). Unfortunately, the latest version of pqR is based on GNU R 2.15.0 (M. Neal, 2019). This makes geospatial packages like `sf` not compatible with it since `sf` needs R version > 3.3.0.(Pebesma et al., 2019). Other examples of R packages that are no compatible are `stars`, `RQGIS`, and the latest of `sp`. `sf` is a mandatory package to run the geospatial workflow on the benchmark step. Without this package, it is not possible to run the benchmark. Therefore, the geospatial Docker image for pqR is not possible to create although it is possible to create the vanilla version.

Fedora

Since it is not possible to install geospatial packages on pqR on Debian, the same situation applies for pqR on Fedora. Because of this reason, the Docker image for pqR on Fedora is not created and marked as *stop* in the final result.

Arch Linux

Since it is not possible to install geospatial packages on pqR on Debian, the same situation applies for pqR on Arch Linux. Because of this reason, the Docker image for pqR on Arch Linux is not created and marked as *stop* in the final result.

4.1.6 TIBCO Enterprise Runtime for R (TERR)

Debian

Although TERR is not free and open, TIBCO provides TERR for free with a Developer Evaluation license. This license has a limitation which prevents publishing the Docker image of TERR[16]. With this limitation, this research only provides Dockerfile to build the Docker image that needs the installer that can be obtained from the TIBCO website. TERR is successfully installed for Debian and Ubuntu. Unfortunately, there is a problem when R geospatial packages are tried to be installed. If there is already a package installed, it is not possible to re-install the package. For example, package `vctrs` is already installed from another R packages installation, if we want to install `RSQLite` which depends on `vctrs` it will turn an error.

Another problem found is related to the `rJava` package. An error message suggests running `R CMD javareconf` for successful `rJava` installation. Unfortunately, this `javareconf` is not available in TERR.

From this problem, it is not possible to install geospatial packages that have the same R package dependency. In the end, it is possible to create a Docker image for TERR on Debian but it is not possible to create its geospatial Docker image.

[16] `https://tap.tibco.com/storefront/product-view.ep?pID=15307`

Fedora

Since it is not possible to install geospatial packages on TERR on Debian, the same situation applies for TERR on Fedora. Because of this reason, the Docker image for TERR on Fedora is not created and marked as *stop* in the final result.

Arch Linux

Since it is not possible to install geospatial packages on TERR on Debian, the same situation applies for TERR on Arch Linux. Because of this reason, the Docker image for TERR on Arch Linux is not created and marked as *stop* in the final result.

4.2 Benchmark

This section is divided into three parts. The first one is a brief explanation of the benchmarking tool that is created and used in this research. The second part is the result and analysis of the baseline benchmark. And the last one is the result and analysis for the geospatial workflow benchmark.

4.2.1 Benchmark Tool (`altRnative`)

A new R package name `altRnative` is created to provide functionalities to do benchmark for R script across different Docker image (Sunni & Nüst, 2020a). `altRnative` uses `microbenchmark` (Mersmann et al., 2019) for running the benchmark process and `stevedore` (FitzJohn, 2019) for managing the Docker image and containers. This package is available in the GitHub repository, in the following link: `https://github.com/ismailsunni/altRnative`.

In the current version, `altRnative` only supports the successful Docker image created in this research as shown in table 4.2. Users can choose which platform and R implementation that will be used in the benchmarking. The R script target for the benchmark can be provided to the `code` parameter. It can accept a raw string that represents R code or an `expression`. For running R code from a file, it can use the `source` method. An example of a running benchmark can be seen below.

```
baseline_benchmark_result = benchmarks_code(
  code = "1 + 1",
  r_implementations = c('gnu-r', 'mro'),
  platforms = c('debian', 'ubuntu', 'fedora', 'archlinux'),
  times = 10
  )
```

After the benchmark result is obtained, it can plot using `microbenchmark`'s default plot or box plot. Other than that, `altRnative` provides its own box plot function for most common usage. The sample plot can be seen in section 4.2.2 and 4.2.3.

Table 4.3: Baseline Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	1.27	1.32	1.31	1.43
gnu-r on fedora	1.16	1.27	1.26	1.57
gnu-r on archlinux	1.21	1.25	1.25	1.28
mro on ubuntu	1.30	1.36	1.37	1.41
mro on fedora	1.22	1.32	1.32	1.42

Another useful feature of `altRnative` in calculating the elapsed time ratio between the benchmark result using `normalize_benchmark_result` function. This function gives the ratio for each benchmark result compared to the mean of selected Docker image (implementation and platform.). From this result, it is easier to see the comparison between Docker images. From this result, it also can be plotted using the default box plot method, or directly using `normalize_benchmark_boxplot` from `altRnative` that calculating the ratio internally.

4.2.2 Baseline Benchmark

The baseline benchmark is used to check the time spent for running the Docker container and the R inside it. To do it, a very simple R script (`1 + 1`) is run 10 times for each Docker image. The summary can be seen at table 4.3. It's clear that there is not so much difference between Docker images for this baseline benchmark. The distribution of the baseline benchmark is shown in box plot figure 4.1. There are two outlier observations, but in general, the result is quite similar within the 0.1-second range for the mean.

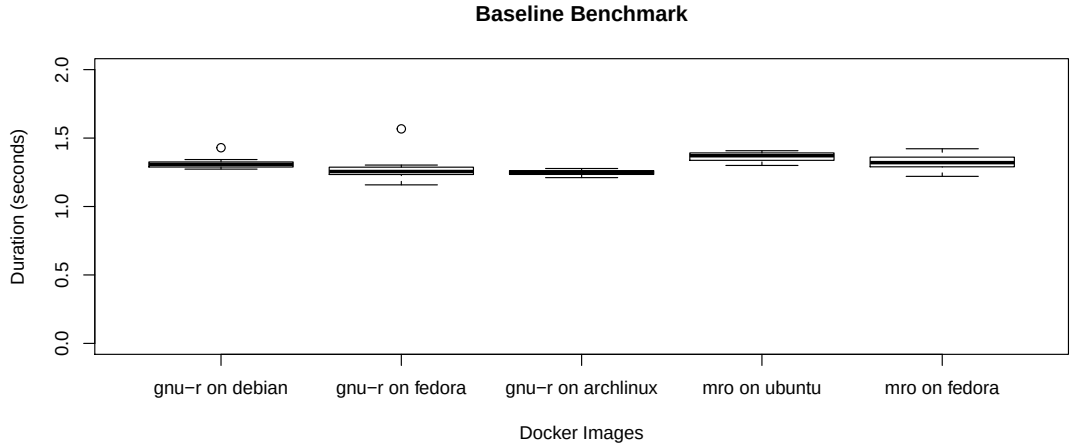

Figure 4.1: Baseline Benchmark

Table 4.4: SDSR Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	26.7	27.1	27.1	27.3
gnu-r on fedora	27.4	27.7	27.7	28.0
gnu-r on archlinux	27.9	28.1	28.0	28.4
mro on ubuntu	33.6	33.7	33.7	34.0
mro on fedora	33.2	33.4	33.4	33.8

Table 4.5: Normalized SDSR Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	0.987	1.00	1.00	1.01
gnu-r on fedora	1.014	1.02	1.02	1.03
gnu-r on archlinux	1.030	1.04	1.04	1.05
mro on ubuntu	1.241	1.25	1.25	1.26
mro on fedora	1.226	1.24	1.24	1.25

4.2.3 Geospatial Workflow Benchmark

For running geospatial workflow, the code from the SDSR book is extracted using an R script to avoid overhead time in knitting the book. This research does not use the original version of SDSR but uses a modified one. A modification is needed to avoid downloading big size data from `starsdata` (SDSR chapter 4) (Pebesma, 2019a) and Air Quality data (SDSR chapter 16). Other modifications are minor like adding needed libraries per chapter and remove evaluation for an unavailable library. The modified SDSR data can be found in Edzer Pebesma et al. (2020) or this Github repository[17].

The R script to run the benchmark can found in the vignette of `altRnative`. While the raw result of the benchmark can be found in this thesis Github repository[18].

The result of this benchmark can be seen in table 4.4 and its distribution in box plot figure 4.2. In this result, there is no outlier for all Docker images. We can see also that GNU R has better compared (~27-28 seconds) to MRO (~33 seconds) regardless of the operating system. For a better view of the speed comparison, a normalized time is created by using the mean of time for GNU R on Debian as the baseline. The result is shown in table 4.5. It's clear that MRO needs 1.25x time compared to GNU R.

[17] `https://github.com/ismailsunni/sdsr/tree/v1.0.0`
[18] `https://github.com/ismailsunni/MasterThesis/`

Table 4.6: Duration per Chapter for All Docker Images

Chapter	Mean
01-hello.R	4.99
02-Spaces.R	1.35
03-Geometries.R	2.85
04-Raster-Cube.R	26.01
05-GeomManipulations.R	2.31
06-Attributes.R	2.69
07-ReferenceSystems.R	1.75
08-Plotting.R	2.05
09-BasePlot.R	2.80
10-Ggplot2.R	3.93
98-rbascis.R	1.71

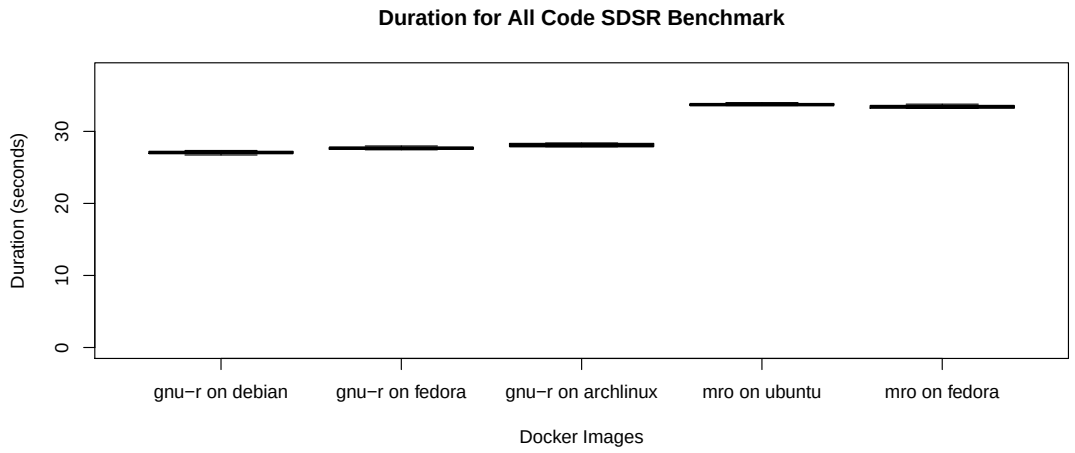

Figure 4.2: Duration for All Code SDSR Benchmark

It also can be check which chapter of the book that takes the most time to run in table 4.6. Chapter `04-Raster-Cube.R` takes the most of the time compared to other chapters with 26.01 seconds. From here we can examine the benchmark result of chapter `04-Raster-Cube.R`.

Based on table A.4 the difference between the Docker images are not big. It is more clear in the normalized benchmark table 4.8 that MRO needs around at most 1.2x time compared to the GNU R one. There is also no outlier in the benchmark `04-Raster-Cube.R` as shown in box plot figure 4.3. Full result for benchmark per chapter can be found in appendix A.

Table 4.7: Chapter 04 Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	23.9	24.1	24.1	24.3
gnu-r on fedora	24.3	24.6	24.6	24.7
gnu-r on archlinux	24.1	24.3	24.3	24.4
mro on ubuntu	28.3	28.6	28.6	28.8
mro on fedora	28.2	28.5	28.5	28.7

Table 4.8: Normalized Chapter 04 Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	0.992	1.00	0.999	1.01
gnu-r on fedora	1.011	1.02	1.021	1.02
gnu-r on archlinux	0.999	1.01	1.008	1.01
mro on ubuntu	1.176	1.19	1.188	1.20
mro on fedora	1.171	1.18	1.185	1.19

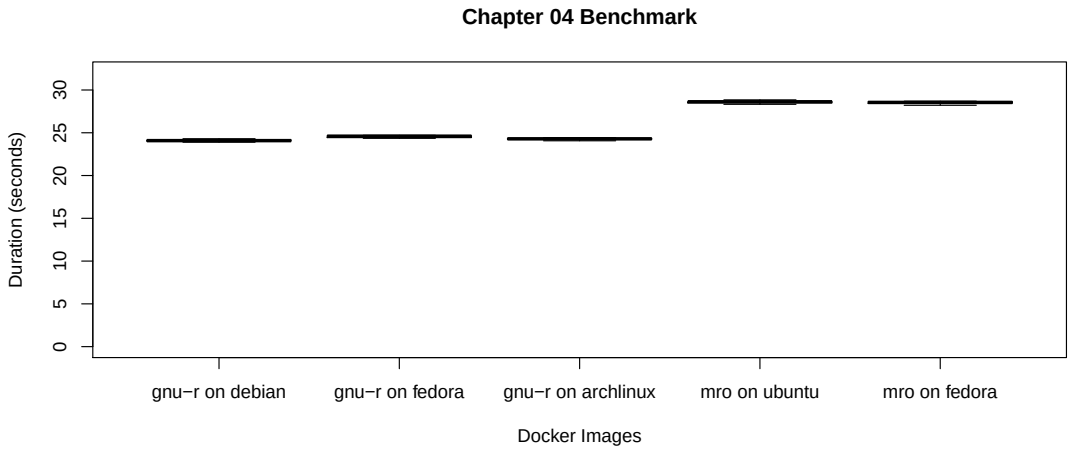

Figure 4.3: Chapter 04 Benchmark

Chapter 5

Discussion

This chapter contains a discussion about the result of the research. It is divided into three parts. First, the common problems that are found when creating a Docker image for various R implementation and platform are compiled. The result of the benchmark is discussed in the second part. The last part discusses the limitation and the possible improvement of the research.

5.1 Common Problems for R Implementations and Platform

Based on the result in chapter 4, the problem found when creating a Docker image for different R implementation on the different platforms can be grouped into three kinds of problems. The first one is related to system dependencies, second is related to unsupported implementation, and the last one is about error or problem in running time. An example of each common problem can be seen in table 5.1. Below for each common problem is discussed.

5.1.1 System Dependencies

In this research, system dependencies are taken from sysreqsdb which provide a list of system dependencies for each R package based on the metadata. This database is almost complete, there are only two outdated system dependencies for Fedora 32 as shown in table 5.2. Some Docker image needs to install another system dependencies that are not available in the base image for example `gfortran`, `sqlite`, and `sqlite-devel`. But in this case, it's more on that the system dependencies are not mentioned in the R packages. This kind of error is relatively easy to solve for example by checking the new name of the package and try to install it again. Another kind of problem is related to the platform. Platforms do not provide the same support for system dependencies. For example, in Oracle Linux 7 does not have GDAL by default. It needs to add Extra Package for Enterprise Linux (EPEL) to get GDAL and other packages("EPEL - Fedora Project Wiki," 2019). It turns out, GDAL from EPEL is very old (version 1.11). In this case, GDAL must be built from the source.

Table 5.1: Common Problems Example

System Dependencies	**Unsupported Implementation**	**Running Time Problem**
Outdated or missing dependencies from sysreqsdb, for example, v8, sqlite, gfortran.	pqR is based on GNU R version 2.15.1 thus it does not support ‘sf‘	Error reinstalling R package on TERR
System dependencies not available on the platform. For example, GDAL is too old on Oracle Linux.	Renjin needs to download every time it tries to load a package	Error loading gpkg with ‘sf‘ for FastR
System dependencies are installed but not usable. For example Udunits on Renjin	TERR does not have javareconf command/file	
System dependencies are too new for the R packages. For example ‘sf‘ in MRO 3.5.3		

Table 5.2: Difference between SysreqsDB and Fedora 32

Library	**SysReqsDB**	**Fedora 32**
libproj	proj-devel proj-epsg proj-nad	proj-devel proj
v8	v8-314-devel	v8-devel

It applies for other packages also which need the newer version and it is not always easy to build from source.

Reversely, some packages need older versions of system dependencies. For example, `sf` on MRO 3.5.3 needs GDAL version 2.x, not 3.x. Fedora 32 has GDAL version 3 while Fedora 30 has GDAL version 2. This happens because `sf` on MRO 3.5.3 snapshot is the old version (v 0.7.3) which depends on GDAL 2.x. To solve this kind of problem, it's either downgrade the packages or wait for the newer release.

After the successful installation of system dependencies, it does not always work properly in the R environment. For example, `proj` is said that it has a configuration error on FastR on Debian/Ubuntu although it can be used properly. This issue has not been solved yet for Debian/Ubuntu.

5.1.2 Unsupported Implementation

The second common problem is the unsupported implementation which refers to the implementation of R itself. This problem occurs because there is no support for a specific feature like in GNU R. For example, pqR is created based on GNU R version 2.15.1. In this case, it is not possible to install R packages which require a newer version of R, for example, `sf` which needs R version 3.3.0 (Pebesma et al., 2019). Another example is Renjin which does not support external native libraries. Many R packages are incompatible with Renjin because of this for example `units` package that needs `udunits2` ("Renjin.Org | units 0.6-3," 2020).

For this kind of problem, there is not so much that the user can do besides hoping that the R implementation will support the features. Another solution is jumping to the R implementation development itself. Unfortunately, this kind of task is not trivial.

5.1.3 Running Time Error

The last kind of error is Running Time Error. This error happens after package installation is finished successfully. This error is hard to find since it can be anything. Based on the SDSR book, the research found three running time error. The first one is the different behavior of `sf` from GNU R with different GDAL and Proj version[1]. There is a change of behavior on GDAL/Proj that is passed to `sf`. The second running time error is on `sf` from FastR on Oracle Linux 7. In this error, `sf` cannot read `gpkg` file[2]. The last is related to TERR which explained in 4.1.6.

Similar to problem with the unsupported implementation, this kind of error is also hard to fix. It is either waiting for the bug to be fixed or fix the bug itself which is no easy task to do.

[1] `https://github.com/r-spatial/sf/issues/1238`
[2] `https://github.com/oracle/fastr/issues/128`

5.2 Benchmark Result

The benchmark result from the experiment is quite clear as previously mentioned in chapter 4.2. The platform difference almost does not give any difference and GNU R performs a little bit better (~1.1x faster) compared to MRO regardless of the platform for geospatial workflow. This not a surprise since MRO's strength is about matrix or vector operation ("The Benefits of Multithreaded Performance with Microsoft R Open . MRAN," 2020) while geospatial workflow in the SDSR books doesn't need a lot of matrix or vector operation.

5.3 Limitation and Recommendations

In this section, the limitation of this research is discussed with the recommendations for improvement that can be done in the future. It also contains some more ideas for future work that can be implemented. There is the part of this section, each one for the Docker Images creation, the benchmarking tool, and the benchmarking process.

5.3.1 Docker Images Creation

Docker image creation takes a long time especially if there is limited information or stumbled on a problem. There are some possible fixes that can be done to create more geospatial Docker images.

1. Downgrade packages in Arch Linux so that it can use GDAL 2.x to support the installation of MRO 3.5.3. Due to time limitation and last-minute knowledge about the unofficial MRO package on AUR("AUR (en) - microsoft-r-open," 2020), this solution has not been checked.
2. Create a Docker image based on Fedora 30 or 32 to install FastR 3.6.1 for updated packages support. This research has only done FastR on Oracle Linux 7.
3. Create a Docker image for Windows to explore the compatibility of geospatial packages and get more performance comparison.

5.3.2 Benchmarking Tool

The altRnative package has shown its capability to perform benchmark across multiple R implementation and platform. But there are some areas that can be improved to get better results. Adding a feature to run a benchmark from the existing Docker container or custom Docker image will make it more flexibility to set up the benchmark process. This feature will also unlock the possibility to run a benchmark not only for geospatial R workflow or non-R script. In this case, altRnative will be a universal benchmarking tool.

Another improvement is adding profiling feature to know which part of the code that spends the most of the time. There is already an R package for profiling, called

`profvis` (Chang, Luraschi, & Mastny, 2019). altRnative can use this R package to get the profiling tool, just like it uses `microbenchmark` to do the benchmarking.

Lastly, this tool can be deployed in a server (cloud) and can be made as a service. This service will help the user to run their R script on different R implementation and platforms without setting up their own infrastructure on their machine. As for the developer, this tool can be used on continuous integration (e.g. Travis) to test their R package continuously.

5.3.3 Benchmarking

In this research, the benchmark is only run for one use case (SDSR book) and only on one machine. This means that the result may be specific to the use case and the machine. Therefore, it is a good idea to run more use cases and on more machines to see whether the result convergence or not.

This research uses the latest version of R implementation. It uses GNU R version 3.6.1 and MRO 3.5.3 which based on GNU R 3.5.3. In this case, there may be already a gap between those two GNU R versions. This version difference also applies to the installed system dependencies' version. For example, GDAL version 2.x on MRO Docker image and GDAL version 3.x or GNU R on Arch Linux Docker image. Therefore, these different versions of R implementation and system dependencies can affect the benchmark result.

Chapter 6

Conclusion

This research explored the possibility of using alternative R implementations and platforms for geospatial R packages. It is shown the information provided by the `sysreqs` database is almost complete only missing update as shown in table 5.2. This research also shows that all targeted R implementations can be installed on all platforms, except the one that is not possible for geospatial R packages. The installation of geospatial R packages itself is not always straight forward. In this research, there are three common problems found in the installation process. It shows that not all alternative R implementation fully supports geospatial R packages.

A benchmarking tool called by altRnative is created to perform a benchmark between a successful Docker image. The result shows that there is not so much difference in terms of speed among the Docker image for geospatial workflow.

In future research in the same area, more Docker images can be created or fix the current failed Docker image. The benchmarking tool can also be extended to make it the universal benchmarking tool not only for R language and geospatial workflow. For a better understanding, the profiling feature can be added to the benchmarking tool.

In the bigger picture, this research's most important contribution is the exploration of R implementation on a different platform for geospatial R packages. This exploration can be useful for R users to choose which R implementation or platform that they can use for their workflow. Another important contribution is the benchmarking tool that can be used for testing or perform a benchmark for user's geospatial workflow.

Appendix A

SDSR Benchmark per Chapter

This appendix contains all the results of the benchmark per chapter in the SDSR book.

Table A.1: Chapter 01 Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	3.69	3.77	3.76	3.85
gnu-r on fedora	3.82	3.87	3.87	3.96
gnu-r on archlinux	4.00	4.10	4.11	4.14
mro on ubuntu	6.61	6.68	6.67	6.76
mro on fedora	6.40	6.51	6.51	6.56

Table A.2: Chapter 02 Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	1.30	1.36	1.36	1.47
gnu-r on fedora	1.25	1.29	1.29	1.33
gnu-r on archlinux	1.25	1.31	1.31	1.36
mro on ubuntu	1.36	1.42	1.40	1.51
mro on fedora	1.31	1.36	1.37	1.43

Table A.3: Chapter 03 Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	2.69	2.76	2.76	2.84
gnu-r on fedora	2.72	2.77	2.78	2.83
gnu-r on archlinux	2.74	2.80	2.80	2.90
mro on ubuntu	2.95	3.06	3.05	3.19
mro on fedora	2.74	2.88	2.88	2.92

Table A.4: Chapter 04 Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	23.9	24.1	24.1	24.3
gnu-r on fedora	24.3	24.6	24.6	24.7
gnu-r on archlinux	24.1	24.3	24.3	24.4
mro on ubuntu	28.3	28.6	28.6	28.8
mro on fedora	28.2	28.5	28.5	28.7

Table A.5: Chapter 05 Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	2.20	2.28	2.29	2.35
gnu-r on fedora	2.05	2.17	2.19	2.24
gnu-r on archlinux	2.11	2.18	2.19	2.25
mro on ubuntu	2.46	2.50	2.50	2.54
mro on fedora	2.39	2.44	2.43	2.51

Table A.6: Chapter 06 Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	2.46	2.54	2.54	2.63
gnu-r on fedora	2.39	2.45	2.46	2.52
gnu-r on archlinux	2.64	2.70	2.69	2.77
mro on ubuntu	2.81	2.89	2.89	2.94
mro on fedora	2.81	2.85	2.84	2.90

Table A.7: Chapter 07 Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	1.63	1.75	1.75	1.80
gnu-r on fedora	1.60	1.65	1.63	1.76
gnu-r on archlinux	1.61	1.65	1.64	1.69
mro on ubuntu	1.83	1.90	1.90	2.00
mro on fedora	1.73	1.79	1.79	1.86

Table A.8: Chapter 08 Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	1.96	1.99	1.98	2.09
gnu-r on fedora	1.87	1.91	1.91	1.95
gnu-r on archlinux	1.86	1.95	1.96	2.04
mro on ubuntu	2.19	2.25	2.26	2.33
mro on fedora	2.04	2.12	2.12	2.20

Table A.9: Chapter 09 Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	2.60	2.69	2.67	2.78
gnu-r on fedora	2.50	2.58	2.59	2.68
gnu-r on archlinux	2.63	2.67	2.68	2.73
mro on ubuntu	2.99	3.06	3.07	3.11
mro on fedora	2.96	3.00	3.00	3.04

Table A.10: Chapter 10 Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	3.60	3.69	3.69	3.75
gnu-r on fedora	3.63	3.71	3.69	3.80
gnu-r on archlinux	3.81	3.89	3.88	3.98
mro on ubuntu	4.23	4.28	4.27	4.36
mro on fedora	4.03	4.08	4.08	4.11

Table A.11: Chapter 98 Benchmark

Docker Image	Min	Mean	Median	Max
gnu-r on debian	1.62	1.68	1.69	1.74
gnu-r on fedora	1.60	1.63	1.63	1.69
gnu-r on archlinux	1.57	1.61	1.62	1.66
mro on ubuntu	1.80	1.83	1.82	1.87
mro on fedora	1.73	1.77	1.76	1.85

References

AUR (en) - microsoft-r-open. (2020). https://aur.archlinux.org/packages/microsoft-r-open/.

BeBetaDriven. (2019). Renjin | The JVM-based interpreter for the R language for statistical computing. Retrieved September 26, 2019, from `https://www.renjin.org/`

BeDataDriven. (2019). Renjin.Org | Packages. Retrieved October 1, 2019, from `http://packages.renjin.org/packages`

Bivand, R. (2019). CRAN Task View: Analysis of Spatial Data. Retrieved from `https://CRAN.R-project.org/view=Spatial`

Chang, W., Luraschi, J., & Mastny, T. (2019). *Profvis: Interactive visualizations for profiling r code*. Retrieved from `https://CRAN.R-project.org/package=profvis`

CRAN - Contributed Packages. (2019). Retrieved September 25, 2019, from `https://cran.r-project.org/web/packages/`

Daniel Miessler. (2019). The Difference Between Fedora, Redhat, and CentOS. *Daniel Miessler*. https://danielmiessler.com/study/fedora_redhat_centos/.

DistroWatch. (2019). DistroWatch.Com: Put the fun back into computing. Use Linux, BSD. https://distrowatch.com/dwres.php?resource=major.

Docker. (2019). What is a Container? *Docker*. https://www.docker.com/resources/what-container.

Edzer Pebesma, Sunni, I., Yadong Liu, Paccioretti, P.,, Robin, Vanderhaeghe, F., & Li, A. (2020). Ismailsunni/sdsr: First release for master thesis. Zenodo. `http://doi.org/10.5281/ZENODO.3671394`

EPEL - Fedora Project Wiki. (2019). https://fedoraproject.org/wiki/EPEL.

FitzJohn, R. (2019, January). Stevedore: Docker Client.

Gossmann, A. (2017, May). 5 ways to measure running time of R code. *0-fold Cross-Validation*. https://www.alexejgossmann.com/benchmarking_r/.

Hadley Wickham. (2019). Performance · Advanced R. Retrieved September 26, 2019,

from http://adv-r.had.co.nz/Performance.html#faster-r

Hornik, K. (2018). R FAQ. Retrieved from https://CRAN.R-project.org/doc/FAQ/R-FAQ.html

Izrailev, S. (2014). *Tictoc: Functions for timing r scripts, as well as implementations of stack and list structures.* Retrieved from https://CRAN.R-project.org/package=tictoc

Kusnierczyk, W. (2012). *Rbenchmark: Benchmarking routine for r.* Retrieved from https://CRAN.R-project.org/package=rbenchmark

Lisic, J. (2017). *Jlisic/R-docker-centos.* Retrieved from https://github.com/jlisic/R-docker-centos (Original work published August 23, 2017)

M. Neal, R. (2019). pqR - a pretty quick version of R. http://www.pqr-project.org/.

Manuel Weiss. (2016, March). How Docker Makes Testing More Efficient. *via @codeship.* https://blog.codeship.com/testing-with-docker/.

Mersmann, O., Beleites, C., Hurling, R., Friedman, A., & Ulrich, J. M. (2019, September). Microbenchmark: Accurate Timing Functions.

Microsoft. (2019). Microsoft R Open: The Enhanced R Distribution . MRAN. Retrieved September 26, 2019, from https://mran.microsoft.com/open

Neal, R. M. (2019). pqR - a pretty quick version of R. Retrieved September 26, 2019, from http://www.pqr-project.org/

Nüst, D. (2019). *Nuest/mro-docker.* Retrieved from https://github.com/nuest/mro-docker (Original work published February 26, 2016)

Nüst, D. (2019a). Nuest/renjin-docker: Dockerfiles for automatic builds on Docker Hub for Renjin, an interpreter for R built on the Java Virtual Machine. Retrieved October 1, 2019, from https://github.com/nuest/renjin-docker

Nüst, D. (2019b, September). Nuest/fastr-docker.

Nüst, D. (2019c, September). Nuest/pqr-docker.

Oracle. (2019a). GraalVM. Retrieved October 1, 2019, from https://www.graalvm.org/

Oracle. (2019b). Reference Manual for R. Retrieved September 26, 2019, from https://www.graalvm.org/docs/reference-manual/languages/r/

Oracle/graalvm-ce - Docker Hub. (2019). https://hub.docker.com/r/oracle/graalvm-

ce.

Pebesma, E. (2019a). *Starsdata: Datasets for stars*.

Pebesma, E. (2019b, December). Edzer/sdsr.

Pebesma, E., Bivand, R., Racine, E., Sumner, M., Cook, I., Keitt, T., ... Baston, D. (2019, September). Sf: Simple Features for R.

R Development Core Team. (2019). CRAN Package Check Results. Retrieved September 25, 2019, from `https://cran.r-project.org/web/checks/check_summary.html`

R-hub/sysreqsdb. (2019, December). The R-hub project of the R Consortium.

Renjin.Org | sf 0.7-4. (2020). http://packages.renjin.org/package/org.renjin.cran/sf.

Renjin.Org | units 0.6-3. (2020). http://packages.renjin.org/package/org.renjin.cran/units/0.6-3.

Sunni, I., & Nüst, D. (2020a). Ismailsunni/altRnative: First release for master thesis. Zenodo. `http://doi.org/10.5281/ZENODO.3671405`

Sunni, I., & Nüst, D. (2020b). Ismailsunni/dockeRs: First release for master thesis. Zenodo. `http://doi.org/10.5281/ZENODO.3671417`

The Benefits of Multithreaded Performance with Microsoft R Open . MRAN. (2020). https://mran.microsoft.com/documents/rro/multithread.

TIBCO. (2019). TIBCO® Enterprise Runtime for R FAQ | TIBCO Community. Retrieved September 26, 2019, from `https://community.tibco.com/wiki/tibcor-enterprise-runtime-r-faq`

Zeileis, A. (2005). CRAN task views. *R News*, *5*(1), 39–40. Retrieved from `https://CRAN.R-project.org/doc/Rnews/`