Journal of Computing and Applied Informatics (JoCAI) Vol. 5, No. 1, 2021| 39 –48

DATA SCIENCE
Journal of Computing and Applied Informatics

Computing the Value of Pi in the Manner of Lambda Function with R Statistical Programming Language

Muhammad Reza Fahlevi[1] and Mohammad Andri Budiman[2]

[1,2]Departemen Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Kampus USU, Medan 20155, Indonesia

Abstract. The value of π is one of the famous mathematical constant, not only to mathematicians, but also to physicists and to engineers. Numerous algorithm is used to compute what is the value of π, but most of programmer do not use lambda function and neglect the aesthetic of their script. This study aims to compute the value π by write it first as infinite series using Riemann sum, and then the computing of it is conducted in R programming language. We involved the role of anonymous function or known as lambda function to make the R code is more beautiful, artistic, and elegant.

Keyword: Anonymous, Area under the curve, Circle, Function, Integral, R, Riemann sum.

Received [date month year] | Revised [date month year] | Accepted [xx Month xxxx]

1 Introduction

Riemann sum is a method to find the area under a given curve $\phi(x)$. If we denote this Riemann sum by letter R. Then R is defined as [1].

$$R = \sum_{i=1}^{\infty} \phi\,(\xi_i)\Delta x$$

Where $\phi(x)$ is defined as the length of bar at the point $x_i = \xi_i$ and Δx is defined as the wide of bar. For example , suppose that we have a quadratic curve $\phi(x) = x^2$, then the area under the curve for $\xi_0 = 0$ to $\xi_n = 4$ if we choose there are 4 bar is

$$R = \sum_{i=1}^{4} \phi\,(\xi_i)\Delta x$$

*Corresponding author at: [1,2]Departemen Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Kampus USU, Medan 20155, Indonesia

E-mail address: muhammadrezafahlevi666@gmail.com (Muhammad Reza Fahlevi), mandrib@usu.ac.id (Mohammad Andri Budiman)

ISSN: 2580-6769 | e-ISSN: 2580-829X | DOI: https://doi.org/10.32734/jocai.v5.i1-5556
Journal Homepage: https://jocai.usu.ac.id

Since the area of each bar is equals, then

$$\Delta x = \frac{\xi_n - \xi_0}{n} = \frac{4-0}{4} = \frac{4}{4} = 1$$

Thus, $\Delta x = 1$area unit. Therefore, the area under the quadratic curve is

$$\begin{aligned} R &= \sum_{i=1}^{4} \phi(\xi_i)\Delta x = \sum_{i=1}^{4} \xi_i^2 \times 1 \\ &= (1^2 + 2^2 + 3^2 + 4^2) \times 1 \\ &= 1 + 4 + 9 + 16 \\ &= 30 \end{aligned}$$

This value is just an approximation of the exact value the area under the curve for $\phi(x) = x^2$. This approximation is getting better by let $n \to \infty$. The graph of this cases [2] is given below.

```
library(ggplot2)
library(ggpubr)

#define the domain for quadratic function
first_quadratic_domain <- seq(0, 4, 1)
second_quadratic_domain <- seq(0, 4, 0.5)
third_quadratic_domain <- seq(0, 4, 0.1)
fourth_quadratic_domain <- seq(0, 4, 0.01)

#define a quadratic function
quadratic <- function(X) return(X**2)

#define the rieamann sum
riemann_quadratic_sum <- function(X) {
    return(sum(X[,2]) * 4 / (length(X[,2]) - 1))
}

first_df_quadratic <- data.frame("x_val" = first_quadratic_domain,
                          "f.x" = quadratic(first_quadratic_domain))
second_df_quadratic <- data.frame("x_val" = second_quadratic_domain,
                          "f.x" = quadratic(second_quadratic_domain))
third_df_quadratic <- data.frame("x_val" = third_quadratic_domain,
                          "f.x" = quadratic(third_quadratic_domain))
fourth_df_quadratic <- data.frame("x_val" = fourth_quadratic_domain,
                          "f.x" = quadratic(fourth_quadratic_domain))

#plot the area
first_plt_quadratic <- ggplot(data = first_df_quadratic, mapping = aes
(x_val, f.x)) + geom_bar(stat = "identity", color = "skyblue", aes(fil
l = f.x)) + geom_line(color = "blue") + ggtitle("n = 4")

second_plt_quadratic <- ggplot(data = second_df_quadratic, mapping = a
es(x_val, f.x)) + geom_bar(stat = "identity", color = "skyblue", aes(f
ill = f.x)) + geom_line(color = "blue") + ggtitle("n = 9")

third_plt_quadratic <- ggplot(data = third_df_quadratic, mapping = aes
(x_val, f.x)) + geom_bar(stat = "identity", color = "skyblue", aes(fil
l = f.x)) + geom_line(color = "blue") + ggtitle("n = 41")
```

```
fourth_plt_quadratic <- ggplot(data = fourth_df_quadratic, mapping = a
es(x_val, f.x)) + geom_bar(stat = "identity", color = "skyblue", aes(f
ill = f.x)) + geom_line(color = "blue") + ggtitle("n = 401")

ggarrange(first_plt_quadratic, second_plt_quadratic, third_plt_quadrat
ic,
          fourth_plt_quadratic, nrow = 2, ncol = 2)
```

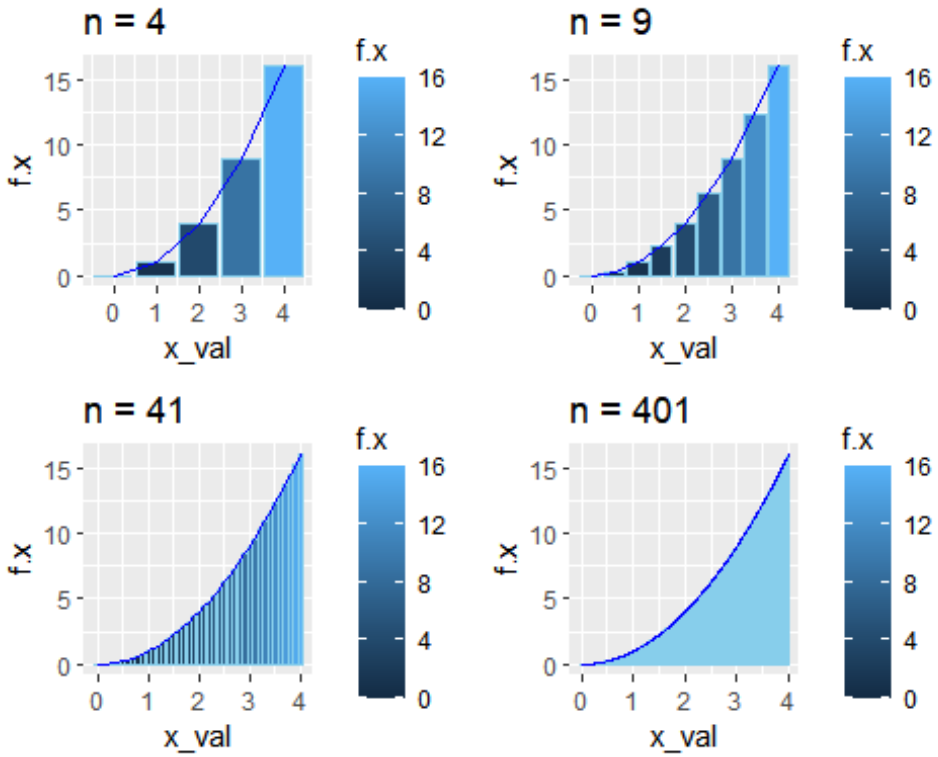

Figure 1 The area quadratic curve for each n = 4, 9, 41, 401.

The area for each n is

```
paste("n = 4, then R = ", riemann_quadratic_sum(first_df_quadratic))

## [1] "n = 4, then R =  30"

paste("n = 9, then R = ", riemann_quadratic_sum(second_df_quadratic))

## [1] "n = 9, then R =  25.5"

paste("n = 41, then R = ", riemann_quadratic_sum(third_df_quadratic))

## [1] "n = 41, then R =  22.14"

paste("n = 401, then R = ", riemann_quadratic_sum(fourth_df_quadratic)
)

## [1] "n = 401, then R =  21.4134"
```

The reader who have studied calculus might be notice that the area under the curve is given by a definite integral between interval [a, b] of the function $\phi(x)\ dx$. That is,

$$\int_a^b \phi\,(x)dx = \sum_{i=1}^{\infty} \phi\,(\xi_i)\Delta x, \qquad \Delta x = \frac{b-a}{n}$$

For $n \to \infty$. Thus, for $\phi(x) = x^2$

$$\begin{aligned}\int_0^4 x^2\,dx &= \frac{1}{3}x^3|_{x=0}^{x=4} \\ &= \frac{1}{3}(4^3 - 0^3) \\ &= \frac{64}{3}\end{aligned}$$

Therefore, the area under the curve is equals to 21.33 unit area.

Numerous study has been done in order to compute the value π, numerous algorithm had been performed. The mathematical constant π is so famous because π is defined as transcendental number, π is not the root of non zero polynomial with rational coefficients, and there are infinite number behind the decimal point of the value π. Monte carlo simulation for calculation π is a method based on statistics and probability [3], and the number of π from billiard point of view is calculate the digit of π based on ideal world of classical physics [4][5]. Wheter Monte carlo simulation or π billiard point of view, both of them is computable and the simulation were beautiful, and yet, not so efficient if we want 8 or more digit the value of π. However, none of them produce a beautiful script of code that return the value or the digit of π. The experiment of the computation the value of π using Riemann sum is conducting in R programming language and using RStudio as an IDE. First, we write the script in the ideal form so it is easy to understand the algorithm, and then we reduce the number of line that's being used by involving anonymous function, $\lambda - \text{function}$.

2 Method

The area of circle is defined as $A = \pi r^2$, r is the radius and π is a constant, that is π = 3.141592… . Now, we try to approximate the value of π by using this formula, but, it is not as simple as $A = \pi r^2$. Firstly, we calculate the value of the area A by using Riemann sum.

Suppose that the circle is centered at point $O(0,0)$ and has the radius $r = 1$. Then, the equation for this circle is defined as

$$x^2 + y^2 = 1$$

The graph of this equation is below

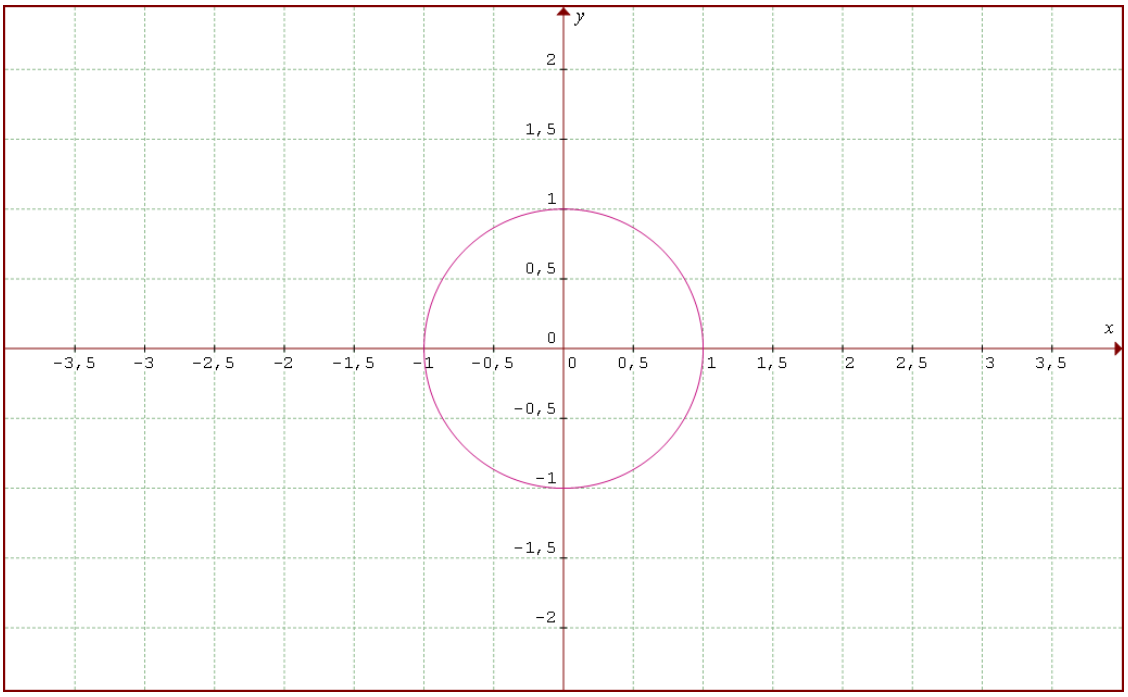

Figure 2 The graph of circle at center point $O(0,0)$

The area of this circle is

$$A = \pi r^2$$

Divided both side by 4, we have

$$\frac{A}{4} = \frac{\pi r^2}{4}$$

Let $R = \frac{A}{4}$, then

$$R = \frac{\pi r^2}{4}$$

R is the area that we want to calculate by using Riemann sum. Then, the area that we want to calculate is a quarter of the actual area of the circle. Our continuous function is $\phi(x) = \sqrt{1 - x^2}$ and the graph of this equation is below

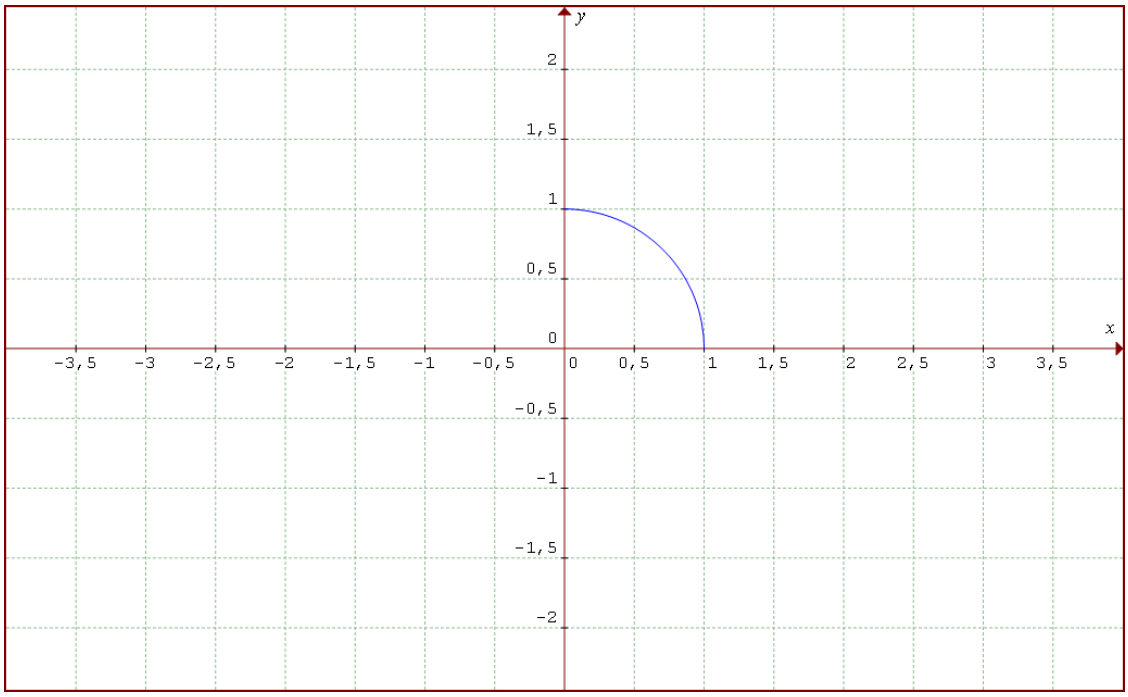

Figure 3 The graph of function $\phi(x) = \sqrt{1-x^2}$

If $x_i = \xi_i$, then by definition, the area under the curve is

$$\int_a^b \phi\,(x)dx = \sum_{i=1}^{\infty} \phi\,(\xi_i)\Delta x, \qquad \Delta x = \frac{b-a}{n}$$

For Δx, if b is the upper bound of the curve and a is the lower bound of the curve, then

$$\begin{aligned} \Delta x &= \frac{b-a}{n} \\ &= \frac{r-a}{n} \\ &= \frac{1-0}{n} \\ \Delta x &= \frac{1}{n} \end{aligned}$$

Thus,

$$\begin{aligned} R &= \sum_{i=1}^{\infty} \phi\,(\xi_i)\Delta x \\ &= \sum_{i=1}^{\infty} \sqrt{1-\xi_i^2} \times \frac{1}{n} \end{aligned}$$

For each value of ξ_i, we have that [6]

$$\begin{aligned}\xi_i &= i \times \Delta x \\ &= i \times \frac{1}{n}\end{aligned}$$

Thus, the area of quarter of circle is given by Riemann sum R

$$\therefore R = \sum_{i=1}^{\infty} \sqrt{1 - \left(i \times \frac{1}{n}\right)^2} \times \frac{1}{n}, \qquad \text{for n} \to \infty$$

Since $\frac{A}{4} = R$, thus

$$\begin{aligned}\frac{A}{4} &= R \\ \frac{\pi r^2}{4} &= \sum_{i=1}^{\infty} \sqrt{1 - \left(i \times \frac{1}{n}\right)^2} \times \frac{1}{n} \\ \frac{\pi \times 1^2}{4} &= \sum_{i=1}^{\infty} \sqrt{1 - \left(i \times \frac{1}{n}\right)^2} \times \frac{1}{n} \\ \frac{\pi}{4} &= \sum_{i=1}^{\infty} \sqrt{1 - \left(i \times \frac{1}{n}\right)^2} \times \frac{1}{n}\end{aligned}$$

Multiply both side by 4

$$\therefore \pi = 4 \times \sum_{i=1}^{n} \sqrt{1 - \left(i \times \frac{1}{n}\right)^2} \times \frac{1}{n}, \qquad \text{for n} \to \infty$$

3 Computation

The Riemann sum method give us that the value of π is defined as the infinite series such that

$$\pi = 4 \times \sum_{\mathrm{i}=1}^{\mathrm{n}} \sqrt{1 - \left(\mathrm{i} \times \frac{1}{\mathrm{n}}\right)^2} \times \frac{1}{\mathrm{n}}, \qquad \text{for n}\to\infty$$

In order to make the equation easy to compute, we rewrite the value of π in term of finite series n and $\mathrm{n} \in \mathbb{N}$, that is

$$\mathrm{R(n)} = 4 \times \sum_{\mathrm{i}=1}^{\mathrm{n}} \sqrt{1 - \left(\mathrm{i} \times \frac{1}{\mathrm{n}}\right)^2} \times \frac{1}{\mathrm{n}}, \qquad \text{for n}\in\mathbb{N}$$

Based on this equation, we are ready to compute the value of π as follows. We define `piriesum` as a function to compute riemann sum $\mathrm{R(n)}$ which take $\mathrm{n} \in \mathbb{N}$ as paramater and return a value pi_approx, that is the approximation of the value of π.

```
piriesum <- function(n) {
    Dx <- 1 / n
    ids <- 0
    for (i in 1:n)
        ids <- sqrt(1 - ((i * Dx) ** 2)) + ids
    pi_approx <- 4 * ids * Dx
    return(pi_approx)
}
piriesum(100000)
```

```
## [1] 3.141573
```

If we include the curly brace in the counting of the number of line that's being used, then our first R codes take 8 line.

In this section, we try to reduce the number of line that's being used as minimum as possible. In R programming language, we can input a series or a sequence $s = u_1, u_2, \ldots, u_n$ with the different $u_i - u_{i-1} = d$. R programming language allow us to input a sequence as parameter of the function by using R built-in function seq(<init>, <end>, <iteration>) [7]. By default, if we only input <init = a> and <end = b> as parameter of the function seq, then if a > b then <iteration = -1>, if a < b then <iteration = 1>, and if a = b then function seq will return only one value. For example, seq(0, 6, 2) will return a value 0, 2, 4, and 6 as a vector. R also provide a built-in function sum which return the sum of all element vector. For example, sum(c(1, 2, 3)) will return a value 6. Based on this 2 prebuilt-in function, we can minimize the number of line of the previous code as follows.

```
piriesums <- function(s) {
    Dx <- 1 / length(s)
    riesum <- sum(sqrt(1 - (s * Dx) ** 2))
    pi_approx <- 4 * riesum * Dx
    return(pi_approx)
}

S <- seq(1, 100000) # will generate a sequence 1, 2, … , 100000
piriesums(S)
```

```
## [1] 3.141573
```

The number of line that being used for our R code is equals to 6. We have reduced to 8 - 6 = 2 lines.

We still being able to reduce the number of line by directly write Dx to 1 / length(s). And the code will looks like as follows.

```
piriesumss <- function(s) {
    riesum <- 4 * sum(sqrt(1 - (s * 1 / length(s)) ** 2)) * 1 /length(s)
    return(riesum)
}
piriesumss(S)
```

```
## [1] 3.141573
```

As we can see that the number of line that's have been reduced is equals to 4. But, it's not a good practice since the term 1 / length(s) repeated twice.

Here is the lambda function take a role. R is a high level programming language. R provide a lambda like syntax, lambda function is an anonymous function and it's useful to make our code simple and elegant. Here is a few example of lambda function in R.

```
(function(x) x + 1)(3) # R directly execute it

## [1] 4

increment <- function(x) x + 1 # assign it to a variable
increment(16) # Now, increment is a function which x as paramaters

## [1] 17
```

Suppose that `alriesum` is a lambda function which take 2 parameter `s` and the length of vector `s`, `s` is a sequence and it's a vector. Then, `alriesum` is a lambda function which return the value π and the code of `alriesum` is below

```
# R directly execute the following code
(function(s, Dx = 1 / length(s)) 4 * sum(sqrt(1 - (s * Dx) ** 2)) * Dx)
(S)

## [1] 3.141573

# Assign it to a variable alriesum
alriesum <- function(s, Dx = 1 / length(s)) 4 * sum(sqrt(1 - (s * Dx) *
* 2)) * Dx
alriesum(S) # Now, alriesum is a lambda function which only take 1 par
ameter

## [1] 3.141573
```

4 Results and Discussion

We have reduce the number of line of our R code from 8 to 1, and if we want to compute directly without assign it to an object or a variable, then our R code become

```
S <- seq(1, 100000000)
(function(s, Dx = 1 / length(s)) 4 * sum(sqrt(1 - (s * Dx) ** 2)) * Dx)
(S)

## [1] 3.141593
```

If we want to assign it to a variable, for example `alriesum`, then our R code become

```
alriesum <- function(s, Dx = 1 / length(s)) 4 * sum(sqrt(1 - (s * Dx) *
* 2))
```

and we have a lambda function `alriesum`, which it take sequence as its parameter, that is $1,2,\ldots,n$. We need to take a note that just because the number of line of code is a few / little, it doesn't mean that the program will run faster. We compute the value of π in the manner of $\lambda-$function as the pursue of the simplicity, beauty, and elegance. In fact, our $\lambda-$function to calculate the value of π will start to calculate slowly if we input a sequence $1,2,\ldots,n$ for $n \geq 100000000$. Since our $\lambda-$function to calculate the value of π is based on Riemann sum, then our $\lambda-$function is still more efficient than calculate the digit of π using pool simulation, that

is, in order to produce the digit 314, then the mass of M (kg) we need is $100^3 = 1000000$. Meanwhile, calculation the value of π using Riemann sum $R(n)$ give us that $R(n = 1000000) = 3.141591$.

5 Conclusion

In the term of the number of line that's being used, we consider that this paper have produced a shortest code that calculate the value of π. We have computed the value of π based on Riemann sum, and write it in the manner of $\lambda - \text{function}$ with statistical programming language R. We write it in the pursue of simplicity, beauty, and elegance. However, we found that our code tend to run slowly if we input a sequence $1,2,\ldots,n$ for $n \geq 100000000$.

REFERENCES

[1] J. Stewart, D. K. Clegg, and S. Watson, *Calculus: early transcendentals*. Cengage Learning, 2020.
[2] H. Wickham, "The tidyverse," *R Packag. ver*, vol. 1, no. 1, p. 1, 2017.
[3] O. Yavoruk, "How does the Monte Carlo method work?," *arXiv Prepr. arXiv1909.13212*, 2019.
[4] H. Bae and J. Chang, "An investigation into computing the digits of pi."
[5] G. Galperin, "Playing pool with $π$ (the number $π$ from a billiard point of view)," *Regul. chaotic Dyn.*, vol. 8, no. 4, pp. 375–394, 2003.
[6] T. W. Körner, *Vectors, pure and applied: a general introduction to linear algebra*. Cambridge University Press, 2012.
[7] W. N. Venables and D. M. Smith, "The R Core Team. An Introduction to R, Notes on R: A Programming Environment for Data Analysis and Graphics, Version 3.6. 3. 2020." 2020.