

WEEKLY ACADEMIC ARTICLE | 02

R cho Data Science

Trần Minh Tiến, Phạm Nhật Hào và Nguyễn Phúc Gia Nghi

*Liên hệ góp ý, hỗ trợ dự án: rdsproject2021@gmail.com

Cộng tác viên hỗ trợ: Lâm Gia Phú, Huỳnh Thị Thu Thảo, Lê Hồ Hoàng Anh, Trần Ngọc Khánh Như, Nguyễn Trọng Nhân, Đoàn Ánh Dương

Tóm tắt

R là ngôn ngữ lập trình thống kê được sử dụng rộng rãi trong nhiều lĩnh vực nghiên cứu dữ liệu, khoa học và toán học. Trong bài viết lần này ta sẽ cùng tìm hiểu lịch sử phát triển, ứng dụng, các cách chạy chương trình R và cùng nhau viết một số đoạn mã R cơ bản.

Từ khóa: R programming, syntax, variable, data structures

1. R là gì?

1.1. Từ ngôn ngữ S đến ngôn ngữ R

S là một ngôn ngữ lập trình cho thống kê được phát triển bởi John Chambers và những người khác tại Bell Laboratories, ban đầu là một phần của AT & T Corp. Mục tiêu của S như John Chambers mong muốn là "từ ý tưởng có thể viết phần mềm một cách nhanh chóng, đơn giản và hiệu quả". S được khởi xướng vào năm 1976 như một môi trường thống kê nội bộ dựa trên các thư viện Fortran, những phiên bản S đầu tiên thậm chí còn không có các hàm để lập các mô hình thống kê.

Vào năm 1988, hệ thống được viết lại bằng C và bắt đầu tương đồng với ngôn ngữ S mà chúng ta biết ngày nay (phiên bản thứ 3 của ngôn ngữ này). Bản thứ 4 được phát hành năm 1998 và là phiên bản chúng ta đang sử dụng ngày nay được trình bày trong cuốn *Programming with Data*[1] của John Chambers.

Vào đầu những năm 90, con đường phát triển ngôn ngữ S đã trở nên quanh co và vào năm 1993, Bell Labs đã cấp cho StatSci(sau này là Insightful Corp) giấy phép độc quyền để phát triển và thương mại. Năm 2004, Insightful đã mua lại S với giá 2 triệu đô la và sau đó vài năm đã phát triển và bán ngôn ngữ S với tên S-PLUS với các tính năng mới (chủ yếu là giao diện người dùng) trên đó. Năm 2008, Insightful được TIBCO mua lại với giá 25 triệu đô la và cho đến ngày nay, TIBCO là chủ sở hữu và phát hành độc quyền ngôn ngữ S.

Các nguyên tắc cơ bản trong ngôn ngữ S đã không thay đổi đáng kể từ khi John Chambers xuất bản "The green book" vào năm 1998, và cũng trong năm đó S đã giành được "Association for Computing Machinery's Software System Award" là một giải thưởng danh giá trong lĩnh vực khoa học máy tính.

Hiểu triết lý chung của S cần thiết với cả người đang tìm hiểu S và R vì nó cũng chính là tiền đề tạo nên hai ngôn ngữ này. S nhìn thoáng qua có thể là viết tắt của từ statistics nghĩa là thống kê và đó cũng chính là nguồn gốc của ngôn ngữ S. S không xuất phát từ bất cứ nền tảng ngôn ngữ lập trình nào như truyền thống, điều đó khiến cho những người làm trong lĩnh vực lập trình cảm thấy rất kỳ lạ và khó hiểu. Các nhà phát minh đã tập trung làm cho việc phân tích dữ liệu với S trở nên dễ dàng nhất có thể và S có thể được sử dụng bởi tất cả mọi người, không yêu cầu kỹ năng lập trình hay tư duy lập trình cao.

Trong "Stages in the Evolution of S", John Chambers có viết: "*Chúng tôi muốn người dùng có thể bắt đầu trong một (môi trường tương tác), nơi họ không ý thức rằng bản thân đang lập trình. Cho đến khi nhu cầu của*

họ trở nên rõ ràng và có một mức độ thông hiểu nhất định, họ có thể dần chuyển sang việc lập trình, khi đó khía cạnh ngôn ngữ và hệ thống sẽ trở nên quan trọng hơn.”¹

Phần quan trọng ở đây chính là quá trình chuyển đổi sang việc lập trình. Điều đó muốn nói lên rằng S là một ngôn ngữ phù hợp với tất cả mọi người, từ người không có nhu cầu lập trình đến những người có kỹ năng lập trình tốt. Nói theo mặt kỹ thuật, họ cần tạo ra một ngôn ngữ phù hợp để phân tích dữ liệu tương tác (dựa trên dòng lệnh nhiều hơn cũng như để viết các chương trình dài hơn (giống như các ngôn ngữ lập trình truyền thống khác).

Ngày nay, khi bạn nhìn thấy một công cụ cực kỳ mạnh mẽ hỗ trợ cho thống kê, dự báo, trực quan hóa dữ liệu được các nhà khoa học dữ liệu hay các lãnh đạo quản lý dự án sử dụng rất phổ biến, khả năng cao đó chính là R hoặc là một cái gì đó được xây dựng từ R. Một đặc điểm thú vị của R chính là cú pháp của nó tương tự S-PLUS, mà S hay S-PLUS như đã nói ở trên là một công cụ mạnh mẽ, hiệu quả nhưng lại dễ sử dụng cho tất cả mọi người (tuy cả hai gần như cùng cú pháp nhưng sẽ có một số khác biệt về cách sử dụng)². R là ngôn ngữ lập trình thống kê có mã nguồn mở và miễn phí, do đó nó được sử dụng một cách rộng rãi từ các biểu đồ thống kê truyền thông, đến biểu đồ kinh tế, dự báo thời tiết, các dự án thống kê học máy... đã giúp điều hành hệ thống kinh tế và xã hội của toàn thế giới.

R được ra mắt lần đầu vào đầu những năm 90 bởi Robert Gentleman và Ross Ihaka, cả hai đều là những thành viên của đại học Auckland. Những kinh nghiệm của Robert Gentleman và Ross Ihak được đăng trên "Tập chí Thống kê Tính toán và Đồ thị"[3].

Năm 1995, Martin Machler đã có một đóng góp lớn khi thuyết phục Ross và Robert sử dụng "GNU General Public License" để tạo ra phần mềm R miễn phí. Điều này rất quan trọng vì nhờ đó mà R là ngôn ngữ mã nguồn mở tạo ra tiền đề cho nhiều phần mềm, công cụ được xây dựng dựa trên R một cách hoàn toàn miễn phí.

Năm 1997, R Core Group được giới thiệu với một số thành viên có liên kết với S và S-PLUS. Hiện tại nhóm này là nhóm cốt lõi quản lý mã nguồn của R và chỉ có thể kiểm tra các thay đổi đối với source code chính của R, phần còn lại có thể được đóng góp bởi mọi người trên khắp thế giới dưới dạng thêm tính năng mới hoặc sửa lỗi. Vào năm 2000, R 1.0.0 đã được phát hành thành công ra thế giới.

Ngày nay, R chạy được trên hầu hết mọi nền tảng máy tính và hệ điều hành. Do nó là ngôn ngữ mã nguồn mở nên ai cũng có thể chỉnh sửa, tối ưu lại cho phù hợp hơn với từng nền tảng mà họ chọn. Một tính năng khá thú vị của R so với các dự án mã nguồn mở chính là được cập nhật bản phát hành thường xuyên, thường là vào tháng 10, các tính năng mới, bản sửa lỗi sẽ được công bố cho mọi người. Trong suốt năm, những bản cập nhật nhỏ vẫn được diễn ra thường xuyên cho thấy sự tích cực, chu đáo của nhà phát triển và phát hành sản phẩm.

R hiện nay vẫn duy trì tốt triết lý ban đầu của ngôn ngữ S là cung cấp một môi trường làm việc tương tác nhưng vẫn đủ mạnh mẽ để phù hợp với mọi người, mọi nhu cầu liên quan đến thống kê, lập mô hình... Theo nhiều cách, một ngôn ngữ có thể thành công bằng cách tạo ra một nền tảng và tất cả mọi người đều có thể đóng góp cho nền tảng đó. Danh sách gửi thư R-help và R-devel đã hoạt động liên tục trong hơn một thập kỷ nay nhằm đáp ứng những nhu cầu cần thiết của mọi người.[4]

1.2. Tại sao lại nên học sử dụng R?

Nếu bạn đã đọc đến đây, có lẽ bạn đã bị cuốn hút bởi charm ngôn của ngôn ngữ S và sau này là ngôn ngữ R. R là ngôn ngữ lập trình thống kê dành cho tất cả mọi người, với R, bạn có thể làm từ tính các phép toán cơ bản, vẽ các đồ thị từ những hàm số cho đến xử lý những bảng dữ liệu lớn hay thậm chí làm những công việc phức tạp như lập các mô hình thống kê, toán học chỉ với một vài câu lệnh đơn giản, dễ học kể cả cho những người chưa biết gì về lập trình.

Một trong những ưu điểm cực kỳ tuyệt vời ở R chính là mã nguồn mở. Bạn hoàn toàn có thể sử dụng R như bao ngôn ngữ lập trình bình thường khác bằng cách tải nó về và cài đặt trên máy tính. Việc R là ngôn ngữ mã nguồn mở và hoàn toàn miễn phí sẽ có những lợi ích sau:

- Ta sẽ luôn có thể thực hiện những phương pháp thống kê mới nhất ngay khi có bất kỳ ai nghĩ ra chúng.
- Các lỗi trong R sẽ được sửa chữa một cách nhanh chóng và minh bạch.
- Hiện nay có nhiều cộng đồng lớn những người yêu thích ngôn ngữ R (useR) và họ có thể giúp bạn giải quyết bất kỳ khó khăn nào đang gặp phải.

¹ Văn bản gốc: "We wanted users to be able to begin in an interactive environment, where they did not consciously think of themselves as programming. Then as their needs became clearer and their sophistication increased, they should be able to slide gradually into programming, when the language and system aspects would become more important."

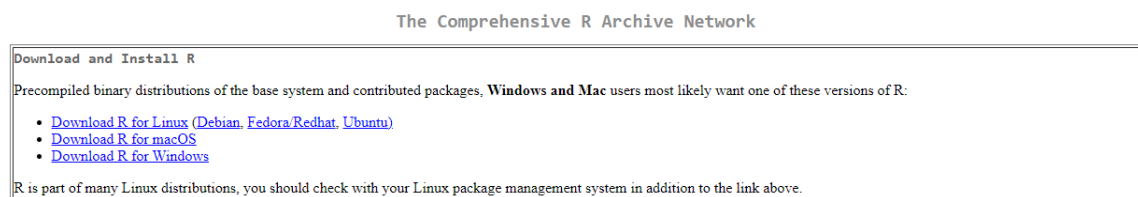
² Thực tế thì ngôn ngữ R về cách hoạt động giống với ngôn ngữ Scheme hơn nhiều so với S.

Việc bắt kỳ ai cũng có thể tạo ra một phiên bản R của riêng mình để phục vụ các mục đích học tập, nghiên cứu khoa học riêng và có thể chia sẻ với mọi người trên toàn thế giới giúp cho R có được một danh sách khổng lồ các thư viện bổ sung. Những người sáng tạo ra nội dung mới sẽ có thể gửi mã code của họ dưới dạng gói (package). Hiện nay có những gói được người dùng viết vô cùng nổi tiếng như gói "pwr" của Stephane Champely chuyên thực hiện các phép phân tích về diện năng hay gói "psych" của APS Fellow William R. Revelle có thể thực hiện bất cứ điều gì từ thống kê mô tả đến mô hình biến trung gian. Như đã liệt kê phía trên, bất cứ khi nào có một phương pháp thống kê mới được công bố, sẽ có các gói công cụ mới và ta hoàn toàn có thể sử dụng R để thực hiện nó.[5]

Bây giờ câu hỏi vẫn chưa được trả lời là bạn nên sử dụng R để làm gì? Tất cả mọi thứ, mọi thứ bạn có thể nghĩ ra trong đầu. R có thể dùng để thực hiện tất cả phép toán, các phép phân tích mô tả, phương trình hồi quy hay bất cứ mô hình nào bạn muốn. Bạn có thể làm việc trên bộ dữ liệu cực lớn mà không cần phải mở excel hay bất cứ phần mềm nào tương tự vì từ việc đọc, xem, làm sạch hoặc chỉnh sửa dữ liệu một cách dễ dàng. R có thể được sử dụng chung với các môi trường gõ văn bản khác như LaTeX, vì vậy bạn hoàn toàn có thể tích hợp luôn kết quả của mình vào các bản báo cáo. Nếu bạn thích sử dụng Microsoft Word, R hoàn toàn có thể đáp ứng với việc tạo các bảng có định dạng APA và xuất chúng dưới dạng tệp .doc. R có thể sử dụng những bộ vi xử lý đa luồng và xử lý song song giúp tăng hiệu suất của mô hình. Bạn có thể sử dụng R để tạo giải một bài toán thống kê trong nháy mắt, R có thể mô phỏng, ngẫu nhiên hóa, lấy mẫu và rất nhiều công việc khác.

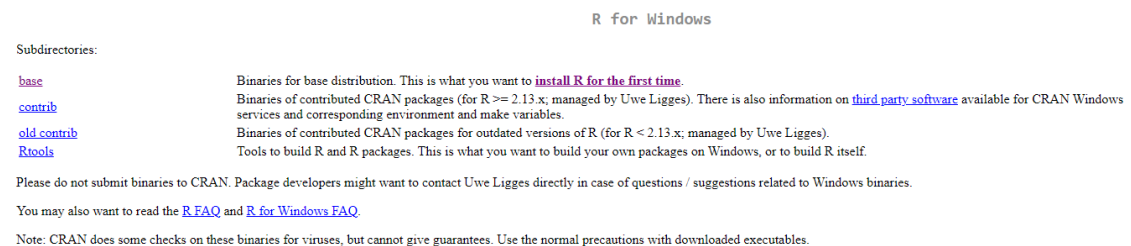
1.3. Cách cài đặt R trên máy tính

Hệ thống R bản gốc và một số thư viện mở rộng có thể được tải về từ "Comprehensive R Archive Network" hay còn được viết tắt là CRAN. Bạn có thể truy cập vào CRAN theo đường link: cran.r-project.org³



1.3.1. Cài đặt trên Windows

Để cài đặt R cho máy chạy hệ điều hành Windows, ta vào CRAN rồi chọn "Download R for Windows"



Chúng ta sẽ có 4 lựa chọn khác nhau:

- **base**: Đây là bản cơ sở của R, nên chọn lựa chọn này nếu đây là lần đầu tiên bạn cài R.
- **contrib**: Đây là nơi bạn có thể tải các gói R mở rộng (phải cài bản base trước) hỗ trợ R từ bản 2.13 trở lên.
- **old-contrib**: Đây là nơi bạn có thể tải các gói R mở rộng cũ hơn hỗ trợ R dưới bản 2.13.
- **Rtools**: Đây là công cụ để bạn có thể xây dựng các gói mở rộng cho R, hoặc xây dựng R theo ý thích của bản thân.

Trong phần hướng dẫn cài đặt mình sẽ sử dụng lựa chọn "base". Khi lựa chọn base bạn sẽ nhìn thấy trang web có phần để tải R về như sau:

³ Toàn bộ hướng dẫn phía dưới được thực hiện vào tháng 03/2022, về sau đường dẫn có thể bị thay đổi hoặc cấu trúc trang web có thể bị thay đổi.

R-4.1.2 for Windows (32/64 bit)

[Download R 4.1.2 for Windows](#) (86 megabytes, 32/64 bit)
[Installation and other instructions](#)
[New features in this version](#)

Khi bạn click chuột vào chỗ "Download R x.x.x for Windows"⁴, R sẽ được tải xuống máy bạn một cách tự động. Có thể có một số máy tính để trình duyệt ở chế độ "Ask where to save each file before downloading" bạn sẽ được hỏi file sẽ được lưu ở đâu sau khi tải về). Sau khi tải về, bạn có thể cài đặt R theo cách cài đặt các phần mềm thông thường trên hệ điều hành của bạn.

1.3.2. Cài đặt trên macOS

Để cài đặt R cho máy chạy hệ điều hành macOS, ta vào CRAN rồi chọn "Download for macOS"

R for macOS

This directory contains binaries for a base distribution and packages to run on macOS. Releases for old Mac OS X systems (through Mac OS X 10.5) and PowerPC Macs can be found in the [old](#) directory.

Note: Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

Package binaries for R versions older than 3.2.0 are only available from the [CRAN archive](#) so users of such versions should adjust the CRAN mirror setting (<https://cran.archive.r-project.org>) accordingly.

R 4.1.2 "Bird Hippie" released on 2021/11/01

Please check the SHA1 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type

```
openssl sha1 R-4.1.2.pkg
in the Terminal application to print the SHA1 checksum for the R-4.1.2.pkg image. On Mac OS X 10.7 and later you can also validate the signature using
pkgutil --check-signature R-4.1.2.pkg
```

Latest release:

R 4.1.2.pkg (notarized and signed)
SHA1-hash: 89e43950b134c322d4dc653220a81a9eaa5 (ca. 87MB)

R 4.1.2.pkg (notarized and signed)
SHA1-hash: 89e43950b134c322d4dc653220a81a9eaa5 (ca. 87MB)

R 4.1.2-arm64.pkg (notarized and signed)
SHA1-hash: 89e43950b134c322d4dc653220a81a9eaa5 (ca. 87MB)

R 4.1.2 binary for macOS 10.13 (**High Sierra**) and higher, **Intel 64-bit** build, signed and notarized package. Contains R 4.1.2 framework, R app GUI 1.77 in 64-bit for Intel Macs, Tcl/Tk 8.6.6 X11 libraries and Texinfo 6.7. The latter two components are optional and can be omitted when choosing "custom install", they are only needed if you want to use the `tcltk` R package or build package documentation from sources.

Note: the use of X11 (including `tcltk`) requires [XQuartz](#) to be installed since it is no longer part of OS X. Always re-install XQuartz when upgrading your macOS to a new major version.

This release supports Intel Macs, but it is also known to work using Rosetta2 on M1-based Macs. For native Apple silicon arm64 binary see below.

Important: this release uses Xcode 12.4 and GNU Fortran 8.2. If you wish to compile R packages from sources, you may need to download GNU Fortran 8.2 - see the [tools](#) directory.

R 4.1.2 binary for macOS 11 (**Big Sur**) and higher, **Apple silicon arm64** build, signed and notarized package. Contains R 4.1.2 framework, R app GUI 1.77 for Apple silicon Macs (M1 and higher), Tcl/Tk 8.6.11 X11 libraries and Texinfo 6.7. **Important: this version does NOT work on older Intel-based Macs.**

Note: the use of X11 (including `tcltk`) requires [XQuartz](#). Always re-install XQuartz when upgrading your macOS to a new major version.

This release uses Xcode 12.4 and experimental GNU Fortran 11 arm64 fork. If you wish to compile R packages from sources, you may need to download GNU Fortran for arm64 from <https://mac.R-project.org/libs-arm64>. Any external libraries and tools are expected to live in `/opt/R/arm64` to not conflict with Intel-based software and this build will not use `/usr/local` to avoid such conflicts.

[NEWS](#) (for Mac GUI)

News features and changes in the R.app Mac GUI

Sau đó lựa chọn phiên bản tùy theo cpu của máy bạn. Nếu là chip intel thì chọn "R-4.1.2.pkg", còn nếu là chip Apple ARM (M1, M2...) thì chọn "R-4.1.2-arm64.pkg".

Trình duyệt của bạn sẽ tự động tải file cài đặt về. Bạn chỉ việc đợi R được tải về rồi cài đặt như cài đặt các phần mềm khác trên macOS.

1.3.3. Cài đặt trên linux

(*Bài viết giả định đang sử dụng Ubuntu 20.04*).

Cài đặt một số thứ cần thiết để thêm một repository mới qua HTTPS:

```
sudo apt install dirmngr gnupg apt-transport-https ca-certificates
software-properties-common
```

Thêm CRAN repository vào danh sách hệ thống:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
E298A3A825COD65DFD57CBB651716619E084DAB9

sudo add-apt-repository 'deb https://cloud.r-project.org/bin/linux/ubuntu
focal-cran40/'
```

⁴ Ở thời điểm bài này được viết là phiên bản 4.1.2.

Cài đặt R:

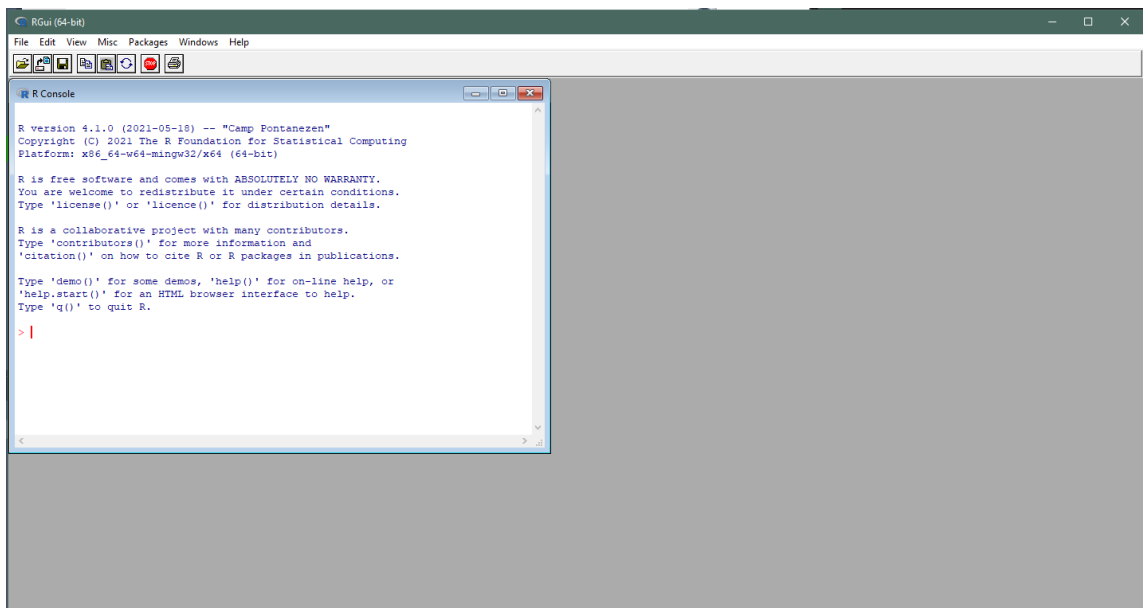
```
sudo apt install r-base
```

2. Một số cách chạy một chương trình R

2.1. R GUI

GUIs là viết tắt của từ Graphical User Interface nghĩa là giao diện người dùng dạng đồ họa. Để dễ hiểu hơn thì chúng ta đang đọc tài liệu này trên máy tính, thông qua một phần mềm, ứng dụng có thể đọc được file PDF. Hãy nhìn xung quanh và bạn sẽ thấy các nút bấm, thanh công cụ, thanh cuộn, thông số của file PDF đang đọc (số trang, số chữ...) đó chính là giao diện người dùng.

Hãy thử tìm kiếm RGui trên công cụ tìm kiếm của hệ điều hành (ví dụ Start ở trên Windows). Đây là công cụ soạn thảo code R được cài sẵn khi ta cài đặt R từ trang chủ CRAN. Khi bạn mở nó lên sẽ thấy được giao diện như sau⁵:



Ta có thể thấy giao diện khá là sơ xài và có nét khá cổ điển. Có vài cách để có thể lý giải điều đó như nhờ giao diện đơn giản và ít tính năng thì RGui có chạy được trên các phần cứng yếu, giao diện đơn giản giúp người mới dễ tiếp cận hơn... Tuy vậy RGui vẫn có đầy đủ những tính năng cơ bản như menubar, toolbar và console.

Bây giờ hãy thử nhập vào console câu lệnh sau: `1 + 1` rồi bấm Enter bạn sẽ nhận được kết quả `[1] 2` chính là kết quả của phép toán `1+1`. Chúng ta sẽ bàn nhiều hơn về các câu lệnh trong R ở các chương sau.

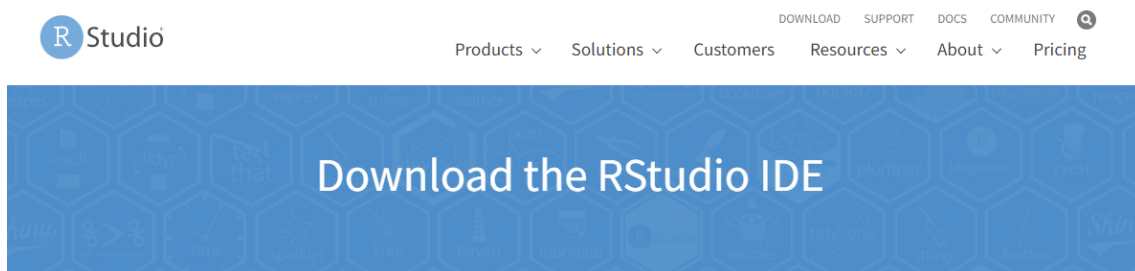
2.2. R Studio

RGui cung cấp cho chúng ta đầy đủ những thứ cơ bản để bắt đầu học và sử dụng R. Tuy nhiên để có thể sử dụng hết được những gì R có thể làm được thì R Studio là một công cụ rất tốt để thay thế. R Studio có giao diện hiện đại hơn có thể tùy chỉnh dễ dàng, hỗ trợ trực quan hóa nhiều tính năng cần thiết và được sử dụng rất rộng rãi trên toàn thế giới.

⁵ Hình ảnh được chụp khi đang sử dụng R 4.1.0, các phiên bản sau sẽ có thể bị thay đổi

2.2.1. Tải và cài đặt R Studio

R Studio có thể được download trên rstudio.com. Khi vào trang chủ của R Studio, bạn nhìn phía trên, chọn Products > R Studio. Trang web sau⁶ sẽ hiện lên.



Cuộn chuột xuống dưới và bạn sẽ nơi để download R Studio, phiên bản miễn phí cung cấp cho bạn đầy đủ những tính năng cơ bản mà bạn sẽ cần để học, sử dụng R cho mục đích nghiên cứu cá nhân. Nếu bạn có tiền và muốn sử dụng thêm một số tính năng nâng cao thì bạn có thể mua phiên bản trả phí. Hiện tại mình đang sử dụng bản miễn phí nên sẽ hướng dẫn bạn tải và cài đặt bản miễn phí.

Để download bạn có thể cuộn chuột xuống ngay phía dưới cái bảng so sánh các phiên bản R Studio. Ở đây bạn có thể phiên bản R Studio cho hầu hết cách hệ điều hành phổ biến hiện nay. Bạn chỉ cần chọn vào phiên bản R Studio muốn tải và sau đó R Studio sẽ được tải về một cách tự động.

Tiếp theo bạn có thể click chuột vào file cài đặt vừa tải về và bắt đầu cài đặt R Studio như cài đặt các phần mềm thông dụng khác.

2.2.2. Giao diện của R Studio

Sau khi cài đặt R Studio bạn hãy mở nó lên, bạn sẽ thấy một cửa sổ tương tự sau (các bạn có thể tùy biến R Studio bằng các theme có sẵn hoặc custom).

Khi mới mở R studio lên, ta có thể thấy màn hình R studio được chia làm 4 cửa sổ nhỏ.

- (1) Cửa sổ **R script**
- (2) Cửa sổ **console, terminal** và **job**
- (3) Cửa sổ **environment, history, connection** và **tutorial**
- (4) Cửa sổ **files, plot, packages, help** và **viewer**

Cửa sổ R script

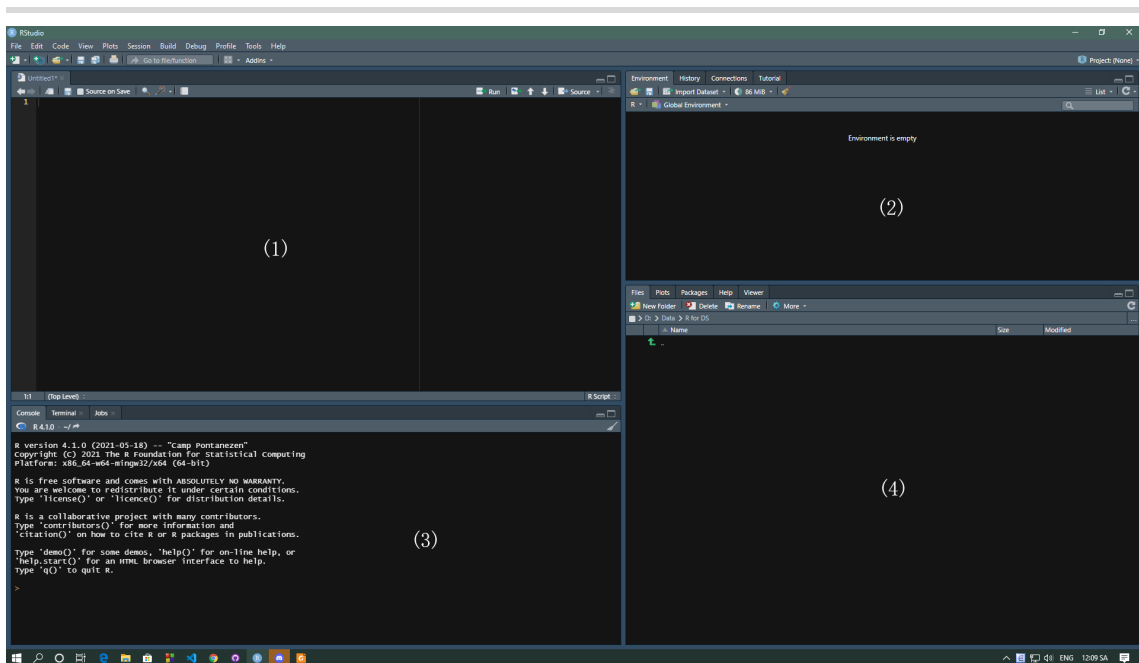
Khi muốn chạy một câu lệnh R, ta sẽ có thể gõ nó lên command line rồi chạy và nhận được kết quả trả về, tuy nhiên cách này có nhiều giới hạn. Ví dụ khi ta có nhiều câu lệnh R cần nhập vào command line để giải quyết một dạng bài toán nào đó và ta sẽ phải dùng nó để giải không chỉ một mà rất nhiều bài toán khác nhau, chắc chắn rằng ta sẽ không thể nhập từng câu lệnh một vào command line và đến bài toán mới ta lại phải chạy lại những câu lệnh đó.

R script chính là giải pháp cho vấn đề nêu trên. R script đơn giản là một file text chứa các câu lệnh R (và có thể sẽ có thêm những ghi chú) được lưu lại và có thể tái sử dụng để giải quyết một bài toán nào đó. Với R script, với một dạng bài toán cụ thể, ta sẽ chỉ cần viết các câu lệnh cần sử dụng vào file R script rồi lưu lại nó, và sử dụng nó mỗi khi cần mà không cần phải gõ lại các câu lệnh.

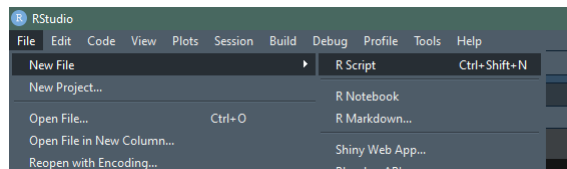
Để tạo một R script mới trong R studio, ta trở chuột đến mục "File" trên thanh điều hướng, chọn **Newfile > R Script** hoặc bấm tổ hợp phím **Ctrl + Shift + N** trên bàn phím. Sau đó bạn sẽ thấy một tab mới xuất hiện trên cửa sổ R Script và đã sẵn sàng cho bạn viết những câu lệnh R vào đó.

```
# Đây là một ví dụ về R Script
a = 12
b = 24
```

⁶ Content được viết vào ngày 07/03/2022, về sau có thể sẽ có thay đổi




Hình 1. Giao diện của R Studio




```
tong = a + b # Tính tong a + b
hieus = a - b # Tính hieu a - b
# In kết quả
a
b
tong
hieus
```

Ta có thể thêm chú thích vào R Script bằng cách sử dụng ký tự #, các ký tự xuất hiện sau ký tự # sẽ là ghi chú và sẽ không được biên dịch trong quá trình chạy.

Sau khi đã có một file R Script, ta sẽ cần lưu lại để có thể tái sử dụng nó. Để lưu một file R Script ta có thể chọn  trên thanh công cụ hoặc sử dụng tổ hợp phím **Ctrl + S**. Sau đó một cửa sổ lưu file hiện lên, bạn sẽ nhập tên bạn muốn vào rồi bấm **save** (Lưu ý bạn sẽ không cần lưu đuôi ".r", khi bạn bấm save đuôi .r sẽ tự động được thêm vào).

Để mở một R Script đã được lưu trong máy, ta trở đến "File" trên thanh điều hướng, rồi chọn **open file** hoặc sử dụng tổ hợp phím **Ctrl + O**, một cửa sổ open file sẽ được hiện lên và bạn có thể chọn mở file R Script bạn cần.

Để chạy một đoạn lệnh R Script trong R studio, bạn có thể dùng chuột để "chọn" đoạn code bạn muốn chạy rồi chọn  hoặc sử dụng tổ hợp phím **Ctrl + Enter** để chạy toàn đoạn code mà bạn đã chọn. Hoặc bạn có thể run mà không chọn khoảng câu lệnh cần chạy, phần mềm sẽ tự động chạy script của bạn cho đến khi xuất ra một kết quả nào đó.

```

1 # Đây là một ví dụ về R Script
2 a = 12
3 b = 24
4 tong = a + b # Tính tong a + b
5 hieu = a - b # Tính hieu a - b
6 # In kết quả
7 a
8 b
9 tong
10 hieu
11

```

```

R 4.1.0 - D:/Data/R for DS/
> # Đây là một ví dụ về R Script
> a = 12
> b = 24
> tong = a + b # Tính tong a + b
> hieu = a - b # Tính hieu a - b
> # In kết quả
> a
[1] 12
> b
[1] 24
> tong
[1] 36
> hieu
[1] -12
>

```

Environment tab

Environment tab là nơi lưu trữ bất kỳ đối tượng, hàm, biến nào bạn tạo ra trong quá trình sử dụng R.

Values	
a	12
b	24
hieu	-12
tong	36

History tab

History tab là nơi lưu trữ những câu lệnh bạn đã chạy trước đó. Những thông tin trong history tab vẫn sẽ được lưu trữ kể cả khi bạn đã tắt RStudio. Để chạy lại những câu lệnh đã chạy trước đó, bạn có thể click đúp chuột vào nó trong history tab.

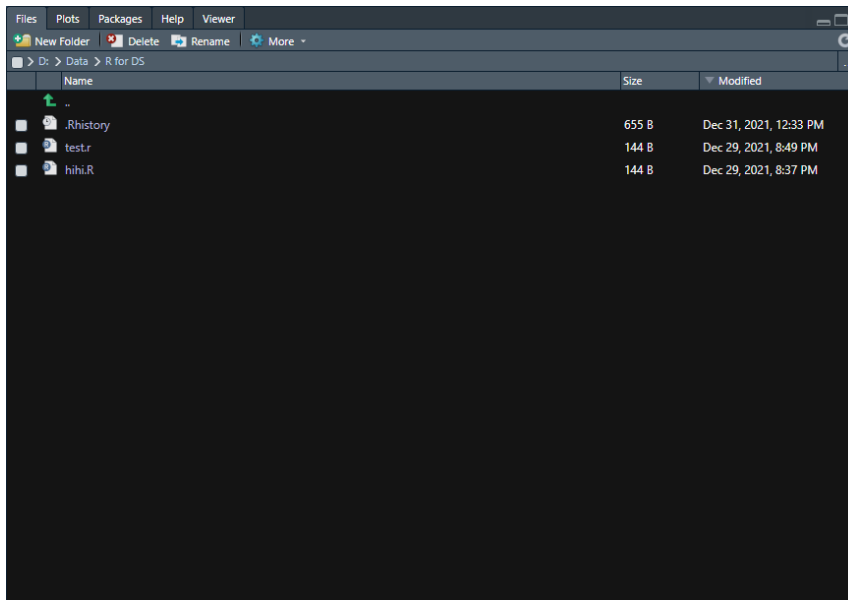
```

a=1
b=2
a+b

```

Files tab

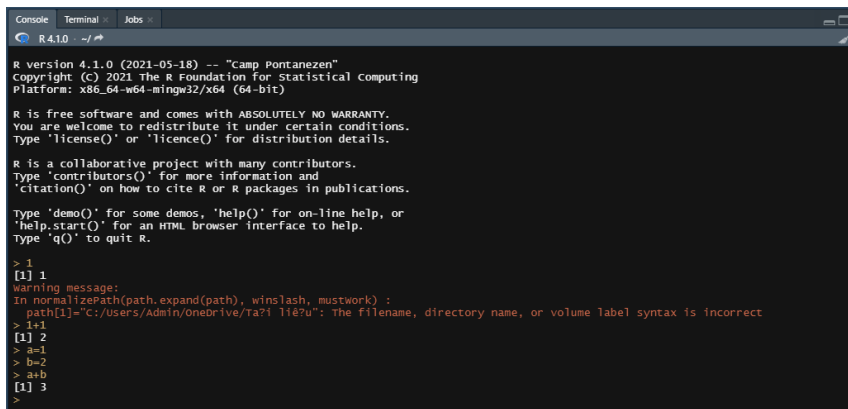
Thông thường Files tab sẽ nằm ở góc phải dưới của RStudio. Files tab hiển thị những tệp của thư mục làm việc hiện tại, bạn hoàn toàn có thể thay đổi được thư mục làm việc. Ngoài ra files tab còn hỗ trợ thêm một số chức năng cơ bản như tạo mới, xóa, đổi tên file, thư mục.



Ngoài ra cùng nằm chung cửa sổ với files tab còn có một số tab nữa như plot dùng để hiển thị biểu đồ (sẽ được nói đến trong các phần sau), packages để xem những packages đã được cài đặt, help và views.

Console

Tương tự như RGUI, RStudio không thể thiếu được console là nơi để thực hiện các lệnh và nhận lại kết quả từ những câu lệnh R. Để thực thi một lệnh R sau khi nhập vào màn hình console, bạn có thể dùng phím Enter.



2.3. RTeX

Như đã đề cập ở phần đầu, R là ngôn ngữ lập trình thống kê được sử dụng phần lớn trong công việc thống kê và phân tích dữ liệu. Các quá trình thống kê và phân tích dữ liệu thường sẽ đi kèm việc lập báo cáo và một trong những công cụ rất phổ biến để viết các báo cáo khoa học chính là $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$. Người ta đã sớm nhận ra sự liên kết thú vị này và sự kết hợp giữa $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ và R là RTeX đã ra đời. RTeX giữ lại gần như toàn bộ tính năng của

L^AT_EX thông thường⁷ và bổ sung thêm R knitr. R knitr là một gói trong ngôn ngữ lập trình R cho phép tích hợp và thực thi các câu lệnh R trong các tài liệu L^AT_EX, LyX, HTML, Markdown, AsciiDoc và reStructuredText. Ta hoàn toàn có thể chuyển một file L^AT_EX sang R_TEX bằng cách thay đổi đuôi của nó từ `.tex` sang `.Rtex`.

Một đoạn script R trong R_TEX được bắt đầu bằng `<>=` và kết thúc bằng ký hiệu `@`.

```
1 \documentclass[10pt]{article}
2 \begin{document}
3   <>=
4   a <- 1
5   b <- 2
6   a + b
7   @
8 \end{document}
```

```
a <- 1
b <- 2
a + b

## [1] 3
```

Vì sử dụng R knitr, ta có thể thêm một số tham số ở giữa `<` và `>` để tùy chỉnh cách hiển thị trong một khối lệnh.

Tham số	Giá trị	Công dụng
message	TRUE/FALSE	bật/tắt hiện các thông tin phụ
warning	TRUE/FALSE	bật/tắt hiện các cảnh báo
echo	TRUE/FALSE	bật/tắt hiện R script
fig.width	số thực	độ rộng của đồ thị
fig.height	số thực	độ cao của đồ thị
fig.align	vị trí	canh lề cho đồ thị
cache	TRUE/FALSE	bật tắt sử dụng bộ nhớ đệm
include	TRUE/FALSE	bật tắt hiển thị tất cả (áp dụng cho cả script và đồ thị)

```
232 \item Bản đồ mức độ hạnh phúc của các vùng, châu lục (có thể
233 đang xét Châu Âu)
234 \begin{figure}
235   <<fig.width=7, fig.height=5, fig.align = 'center', cache = T,
236   echo = FALSE>>=
237   data <- read.csv("world-happiness-report-2021.csv")
238   data2 <- rename(data, region = Country.name)
239   europe <- map_data("world", region = c("Turkey", "Germany",
240     "Albania", "Austria", "Bosnia and Herzegovina", "Belgium",
241     "Bulgaria", "Belarus", "Croatia", "Chipe", "Croacia",
242     "Denmark", "Serbia", "Slovakia", "Slovenia", "Spain",
243     "Switzerland", "Estonia", "Finland", "France", "Greece",
244     "Hungary", "Ireland", "Italy", "Iceland", "Latvia",
245     "Lithuania", "Luxembourg", "Malta", "Norway", "Moldova",
246     "Montenegro", "Netherlands", "North Macedonia", "Poland",
247     "Portugal", "UK", "Czech Republic", "Romania", "Sweden",
248     "Ukraine"))
249   europe <- left_join(europe, data2, by="region")
250   europemap <- ggplot(europe, aes(x=long, y=lat, group = group))
251     + geom_polygon(aes(fill = Ladder.score, color="black"))
252   europemap2 <- europemap + scale_fill_gradient(name = "Ladder
253     score", low = "#CCCCFF", high = "#003366", na.value =
254     "#CCCCC") + theme(axis.text.x = element_text(angle = 90),
255     axis.text.y = element_blank(), axis.ticks = element_blank(),
256     axis.title.y = element_blank(), axis.title.x = element_blank(),
257     rect = element_blank())
258   europemap2
259 @
260 \item Bản đồ mức độ hạnh phúc trên thế giới
261 \begin{figure}
262   <<fig.width=10, fig.height=5, fig.align = 'center', cache = T,
```

⁷ chỉ có một số ít gói L^AT_EX sẽ không hoạt động tốt hoặc gây lỗi trong R_TEX

Trong ví dụ trên có 2 câu lệnh đặc biệt được sử dụng là `%%begin novalidate` và `%%end novalidate`. Điều này cần thiết khi ta sử dụng câu lệnh R có sử dụng một số ký tự trùng với \LaTeX , khi chỉnh sửa trên source code sẽ nhận được thông báo lỗi vì ta đang sử dụng error checking/ validation của \LaTeX (tuy nhiên khi biên dịch sẽ không bị lỗi). Do đó để tránh những lỗi "ảo" phiền phức này, ta sử dụng cặp lệnh `%%begin novalidate` và `%%end novalidate` để tạm thời tắt error checking/ validation của \LaTeX trong đoạn code giữa cặp lệnh đó.

```

1 \documentclass[10pt]{article}
2 \begin{document}
3 <<>=
4 data <- read.csv("world_happiness_report_2021.csv")
5 @
6 %%begin novalidate
7 <<>=
8 data <- read.csv("world_happiness_report_2021.csv")
9 @
10 %%end novalidate
11 \end{document}
```

3. Cú pháp R cơ bản

3.1. Biểu thức

Trong toán học, biểu thức là sự kết hợp hữu hạn các ký hiệu được tạo thành đúng theo các quy tắc phụ thuộc vào ngữ cảnh. Các ký hiệu toán học có thể là những con số, biến số, các phép toán, dấu ngoặc, các ký hiệu logic... Ví dụ dưới đây là một số biểu thức đơn giản:

```

> 1 + 1
[1] 2
> 2^10
[1] 1024
> sqrt(5)
[1] 2.236068
```

Với một số bạn lần đầu học lập trình thì có thể sẽ có cảm giác hơi lạ với các cú pháp toán học trong R. Dưới đây là bảng một số cú pháp toán học cơ bản trong R (bảng chi tiết và đầy đủ hơn sẽ có trong phần phụ lục):

Ký hiệu trong R	Ký hiệu toán học
<code>sqrt(x)</code>	\sqrt{x}
<code>abs(x)</code>	$ x $
<code>x^y</code>	x^y
<code>exp(x)</code>	e^x
<code>floor(x)</code>	$\lfloor x \rfloor$
<code>sin(x)</code>	$\sin x$
<code>cos(x)</code>	$\cos x$
<code>tan(x)</code>	$\tan x$
<code>asin(x)</code>	$\arcsin x$
<code>acos(x)</code>	$\arccos x$
<code>atan(x)</code>	$\arctan x$
<code>log(x, base = y)</code>	$\log_y x$
<code>factorial(x)</code>	$x!$

$$\frac{\sqrt{2 \cdot 4^2 + 1} - 14}{e^3 + 1}$$

```
> (sqrt(2*4^2+1)-14)/(exp(3)+1)
```

```
[1] -0.3915213
```

Hãy sử dụng R để tính một số phép tính sau rồi sử dụng máy tính cầm tay để kiểm tra kết quả:

$$15 + 4 - 2; \quad \sqrt{\sqrt{\sqrt{256}}}; \quad (5 + 2i)^4; \quad \frac{\frac{3}{e^3} - \sqrt{\frac{e^2}{3}}}{6 + \frac{e^2}{\sqrt{2}}}; \quad \frac{\cos \frac{\pi}{3} - \frac{\tan \frac{3\pi}{7}}{21 \sin \frac{\pi}{21}}}{\arctan 0.1417 + \cos \frac{\pi}{5}}; \quad \log_2 3 + \log_3 2.$$

3.2. Kiểu dữ liệu cơ bản

3.2.1. Biến trong R

Biến là tên của vùng nhớ được dùng để lưu trữ dữ liệu và dữ liệu được biến lưu trữ có thể thay đổi trong khi thực hiện chương trình, biến trong R có thể dùng để chứa các giá trị số (cả số thực và số phức), ký tự, ma trận thậm chí là bảng biểu.

R là một ngôn ngữ lập trình động, biến trong R không cần khai báo trước kiểu dữ liệu trước khi sử dụng.

Gán biến là một mảng số nguyên

```
> var.1 = c(0,1,2,3)
[1] 0 1 2 3
```

Biến là một mảng ký tự

```
> var.2 <- c("learn","R")
> var.2
[1] "learn" "R"
```

Chú thích: Hàm c() có tác dụng khởi tạo một mảng (có thể là một mảng số thực hoặc ký tự)

Cách gán giá trị vào một biến trong R

- Sử dụng hàm "assign()"

```
> assign("x", 1)
> print(x)
[1] 1
```

Hàm print(x) có tác dụng in giá trị trong biến ra màn hình, ngoài ra thế thay đổi hàm print(x) thành cat cũng cho kết quả tương tự

```
> assign("x", 1)
> cat(x)
[1] 1
```

- Sử dụng dấu "<-"
(dấu bé hơn < và thêm sau nó một dấu trừ -)

```
> x <- 1
> print(x)
[1] 1
```

- Sử dụng dấu bằng "="

```
> x = 1
> print(x)
[1] 1
```

Các ký tự được dùng để đặt tên biến trong R bao gồm: chữ hoa, chữ thường, số, dấu chấm (.), dấu gạch dưới (_)

Tuy nhiên, đặt tên biến trong một ngôn ngữ lập trình thường có những ràng buộc nhất định và R cũng không

ngoại lệ.

Việc đặt tên biến trong R phải tuân thủ theo các nguyên tắc sau đây:

- Tên biến có các ký tự đặc biệt ngoài ký tự được cho phép (Ví dụ: %, !, ...)

```
> var_name <- 0
[1] Error: unexpected input in "var_name\%<- 0"
```

- Bắt đầu tên biến với chữ số

```
> 3var_name <- 0
[1] Error: unexpected input in "3var_name <- 0"
```

- Bắt đầu với dấu chấm và số kề nhau

```
> .2var_name <- 0
[1] Error: unexpected symbol in ".2var_name"
```

- Bắt đầu với dấu gạch dưới (_)

```
> _var_name <- 0
[1] Error: unexpected symbol in "_var_name"
```

Ngoài ra ta không nên (không được) dùng các keyword của R và các gói (packages) để đặt tên biến

Liệt kê các biến đang có trong môi trường làm việc

Ta có thể liệt kê các biến trong môi trường đang làm việc bằng hàm `ls()`

```
> var.1 <- 0
> var.1
[1] 0
> var.2 <- "Road to Data Science"
> var.2
[1] "Road to Data Science"
> var.3 <- 1.333
> var.3
[1] 1.333
> var.1 <- 0
> var.1
[1] 0
> var.2 <- "Road to Data Science"
> var.2
[1] "Road to Data Science"
> var.3 <- 1.333
> var.3
[1] 1.333
> ls()
[1] "var.1" "var.2" "var.3"
```

Loại bỏ một biến

Ta có thể loại bỏ một biến trong R bằng cách sử dụng hàm `rm()`

```
> a <- 1
> b <- 2
> c <- 3
> ls()
[1] "a" "b" "c"
> rm(a)
> ls()
[1] "b" "c"
```

3.2.2. Các kiểu dữ liệu trong R

1. Kiểu dữ liệu số

Kiểu dữ liệu số học trong R có thể được sử dụng một cách trực tiếp như sau:

```
> 1234
[1] 1234
> 123 + 321
[1] 444
> 2^10
[1] 1024
> 2*3.14
[1] 6.28
```

Đặc biệt ta cũng có thể sử dụng các con số ở hệ thập lục phân trong R với tiền tố 0x. Ví dụ như:

```
> 0x123
[1] 291
> 0x123 + 0x321
[1] 1092
> 0xFF
[1] 255
```

Trong kiểu dữ liệu số, ta có hai kiểu nhỏ thường gặp là số thực và số nguyên (double and integer). Ta có thể xem kiểu dữ liệu của một số x bằng `typeof(x)`:

```
> typeof(1)
[1] "double"
> typeof(3.14)
[1] "double"
```

Trong ví dụ trên có một điểm đặc biệt là số 1 thường được coi như một số tự nhiên lại có kiểu số thực. Trong các ngôn ngữ lập trình bậc cao như python hoặc R, ta không cần phải khai báo kiểu dữ liệu khi khai báo biến nên chương trình sẽ có kiểu dữ liệu mặc định cho các biến số mà ta khai báo là kiểu số thực. Điều này nhằm hạn chế lỗi phát sinh trong quá trình tính toán vì tập số nguyên là tập con của tập số thực và các phép toán với số thực đều có thể áp dụng trên số nguyên.

Vậy nếu ta muốn khai báo số 1 có kiểu là số nguyên "integer" thì sao? Chúng ta có thể ép kiểu, ép kiểu là việc chuyển đổi từ một kiểu dữ liệu này sang kiểu dữ liệu khác, có thể sẽ thay đổi giá trị để phù hợp với kiểu dữ liệu mới. Chúng ta có thể ép kiểu trong R bằng cách sử dụng "as" theo cú pháp: `as(<số cần chuyển>, kiểu dữ liệu mới)`:

```
> as(1, "integer")
[1] 1
> typeof(as(1, "integer"))
[1] "integer"
```

2. Kiểu số phức

Ngoài ra R còn hỗ trợ số phức, số phức trong R có thể được viết dưới dạng <Phần thực> <phần ảo>:

```
> 1 + 2i
[1] 1+2i
> (1 + 2i) + (3 + 4i)
[1] 4+6i
> (1 + 2i) * (3 + 4i)
[1] -5+10i
```

Chúng ta đều biết rằng $i^2 = -1$ và R có thể thực hiện được các phép toán trên số ảo, tuy nhiên nếu ta nhập căn -1 trong console thì sẽ bị lỗi như sau:

```
> sqrt(-1)
[1] NaN
Warning message:
In sqrt(-1) : NaNs produced
```

NaN là viết tắt của Not a Number (không phải là một con số có thể tồn tại). Lý do chính gây nên lỗi trên là do hàm căn bậc hai trong R sẽ giữ nguyên kiểu dữ liệu của tham số. Như ở trên mình có nói khi ta nhập `-1` thì mặc định nó sẽ là số thực và tất nhiên không tồn tại số thực nào là căn bậc hai của `-1`. Để giải quyết lỗi trên ta cần đưa về kiểu số ảo và sẽ không xảy ra lỗi khi thực hiện phép tính trên:

```
> sqrt(-1 + 0i)
[1] 0+1i
```

Trong R ta có 2 cách để truyền tham số cho một biến là sử dụng toán tử `=` hoặc `->`. Trong đó với dấu bằng ta chỉ có thể truyền tham số bên phải cho một biến ở bên trái dấu bằng, còn với dấu `->`, ta có thể truyền theo cả hai chiều với chiều mũi tên chỉ vào biến nhận tham số.

```
> tmp = 3
> tmp
[1] 3
> tmp <- 5
> tmp
[1] 5
> 7 -> tmp
> tmp
[1] 7
```

Ta có thể kiểm tra một biến thuộc kiểu dữ liệu gì bằng lệnh `typeof()` giống như việc kiểm tra dữ liệu với một số trong ví dụ ở phần trước. Ngoài ra, vì R thường được sử dụng trong thống kê nên cũng có thể kiểm tra một dữ liệu là biến định lượng hay biến định tính, để làm việc đó ta sử dụng hàm `class()`.

```
> tmp = 3
> class(tmp)
[1] "numeric"
> typeof(tmp)
[1] "double"
```

3. Kiểu dữ liệu logic

Kiểu dữ liệu logic chứa giá trị hai giá trị Boolean là `TRUE` hoặc là `FALSE`.

```
> logi <- FALSE
> class(logi)
[1] "logical"
> typeof(logi)
[1] "logical"
```

Chúng ta sử dụng hàm `as.logical(data)` để ép kiểu dữ liệu sang dạng logic.

Lưu ý:

- Khi ép kiểu từ số nguyên, số thực hay số phức sang kiểu logic thì trong trường hợp biến mang giá trị bằng 0 thì kiểu dữ liệu logic trả về `FALSE`, ngược lại là `TRUE` trong tất cả trường hợp.
- Kiểu dữ liệu kí tự - được nhắc đến bên dưới - sau khi ép sang kiểu dữ liệu logic thì luôn nhận giá trị `NA`. Ở đây máy tính hiểu `NA` là thiếu giá trị, `NA` \neq `NaN`.

4. Kiểu dữ liệu kí tự

Kiểu dữ liệu kí tự dùng để lưu trữ giá trị dưới dạng kí tự hoặc là dạng xâu, chuỗi - string. Chuỗi trong R có thể là chữ cái, số, kí tự. Để biểu diễn kiểu dữ liệu này, kí tự hoặc xâu được nhập phải ở trong ngoặc đơn hoặc ngoặc kép.

```
> char <- "hello hcmus"
> class(char)
[1] "character"
> typeof(char)
[1] "character"
```

Chúng ta có thể hàm `as.character(data)` để lưu trữ kí tự, một xâu hay ép từ kiểu dữ liệu khác sang dạng kí tự.

```

> char2 <- as.character("hello")
> char3 <- as.character((2+3i)*(4+1i))
> char2
[1] "hello"
> char3
[1] "5+14i"
> class(char2)
[1] "character"
> typeof(char2)
[1] "character"
> class(char3)
[1] "character"
> typeof(char3)
[1] "character"

```

Đến đây ta kết thúc phần kiểu dữ liệu trong R, tiếp theo chúng ta sẽ nói về cấu trúc dữ liệu trong R. Trước đó nếu có câu hỏi thêm về cách mà kiểu dữ liệu hoạt động các bạn có thể mail về địa chỉ project tại mình hoặc comment trực tiếp trong post này để thảo luận.

3.3. Cấu trúc dữ liệu trong R

Trong các ngôn ngữ lập trình khác nhau, chúng ta bắt gặp lập trình viên sử dụng các biến khác nhau để lưu trữ các giá trị dữ liệu khác nhau. Điều này hiển nhiên đúng khi các biến trong cùng kiểu dữ liệu cùng được lưu trong một ô nhớ. Vì vậy cấu trúc dữ liệu là cách duy nhất để sắp xếp dữ liệu và tận dụng hiệu quả trên máy tính.

Khác với những ngôn ngữ lập trình như C hay là Java, R không khai báo các biến dưới dạng kiểu dữ liệu (như số nguyên hay số thực). Thay vào đó, các biến sẽ được gán với R-objects và cơ sở tri thức của R-objects sẽ biến nó thành kiểu dữ liệu. Có nhiều loại R-objects, những cái được sử dụng phổ biến là:

- Vector
- Matrix
- Array
- List
- Data Frames
- Fractor
- String?

1. R Vector.

Vector là cấu trúc dữ liệu cơ bản nhất của ngôn ngữ lập trình R. Nó chia làm 2 phần chính : **Atomic Vector** và **Lists**. Chúng ta có 3 tính chất thông dụng:

- Chức năng chính của cấu trúc dữ liệu đó là gì?
- Độ rộng của cấu trúc dữ liệu đó.
- Thuộc tính của cấu trúc dữ liệu đó như thế nào?

Những yếu tố trên tạo ra các loại cấu trúc dữ liệu khác nhau. Ví dụ, **atomic vector** phải có cùng kiểu dữ liệu trong cùng 1 biến. Ngược lại, mỗi phần tử đại diện cho một **list** có thể có các kiểu dữ liệu khác nhau. Chúng ta sẽ làm rõ điều này ở phần **list** ở dưới. Bây giờ chúng ta sẽ nói về Atomic Vector.

2. Atomic Vectors

Có 4 loại cơ bản trong R Atomic Vectors:

- Kiểu dữ liệu số thực: Numeric Data Type.
- Kiểu dữ liệu số nguyên: Integer Data Type.
- Kiểu dữ liệu kí tự: Character Data Type.
- Kiểu dữ liệu logic: Logical Data Type.

Atomic Vectors chính là kiểu dữ liệu cơ bản của R mà chúng ta đã nói phần trên. Hay ta có thể nói atomic vector chính là kiểu dữ liệu hình thành nên cấu trúc dữ liệu chung của R.

3. R Matrix

Trước hết, chúng ta sẽ thảo luận về ma trận trong R chính xác có nghĩa là gì. Ma trận là một tập dữ liệu có dạng hình chữ nhật và có hai chiều và do đó nó có thể được tạo bằng cách sử dụng đầu vào vector vào hàm ma trận. Ngoài ra, một ma trận là một tập hợp các số được sắp xếp theo một số lượng hàng và cột cố định. Thông thường các con số là số thực. Bằng cách sử dụng chức năng ma trận, chúng ta có thể biểu diễn bộ nhớ của ma trận ở R. Do đó, các phần tử phải có cùng kiểu dữ liệu.

Cú pháp:

```
matrix(data, nrow, ncol, byrow, dimnames)
```

Trong đó :

- **data**: là dữ liệu đầu vào cho ma trận.
- **nrow**: số lượng hàng muốn khởi tạo.
- **ncol**: số lượng cột muốn khởi tạo.
- **byrow**: trả về giá trị TRUE hoặc là FALSE nếu nó là kiểu dữ liệu logic.
- **dimnames**: cho phép đặt tên hàng và cột của ma trận.

```
> mat1 <- matrix(1:4, nrow = 2, ncol = 2)
> mat1
      [,1] [,2]
[1,]    1    3
[2,]    2    4
> mat2 <- matrix(4:7, nrow = 2, ncol = 2)
> mat2
      [,1] [,2]
[1,]    4    6
[2,]    5    7
```

Chúng ta có thể truy cập đến từng phần tử cột thứ m và hàng thứ n như sau:

```
> mat1[1,2] # hàng thu 1 cột thu 2
[1] 3
> mat2[2,1] # hàng thu 2 cột thu 1
[1] 5
```

Chúng ta cũng có thể lấy hết phần tử của một cột hoặc một hàng:

```
> mat1[2, ] #lay het phan tu cua hang 2
[1] 2 4
> mat1[, 2] #lay het phan tu cua cot 2
[1] 3 4
```

Cách để thay đổi ma trận trong R:

a. Gán một giá trị đơn.

Khi gán giá trị vào một vị trí của ma trận trong R sẽ thay đổi giá trị ban đầu của nó và thay thế bằng giá trị mới.

Cú pháp cơ bản là `mat[n,m] <- y` trong đó n và m là các hàng và cột của phần tử tương ứng. với y là giá trị mới cần gán vào ma trận để sửa đổi.

```
> mat1[1,2] <- 5
> mat1[1,]
[1] 1 5
```

Giá trị 3 của biến mat1 đã bị thay thế bằng giá trị 5 ngay tại dòng 1 cột 2.

b. Sử dụng toán tử quan hệ.

Một cách khác để thay đổi một hay nhiều phần tử trong một ma trận trong R là sử dụng toán tử như: `>`, `<`, `==`.

```
> mat1[mat1 == 1] <- 0
> mat1
      [,1] [,2]
```

```
[1,]    0    5
[2,]    2    4
```

Ở đây chúng ta đã sử dụng toán tử `==` để thay thế các giá trị 1 bằng 0. Tương tự chúng ta có thể dùng toán tử `<` để thay thế tất cả giá trị bé hơn 10 thành 0.

```
> mat1[mat1 < 10] <- 0
> mat1
      [,1] [,2]
[1,]    0    0
[2,]    0    0
```

- c. Thêm hàng và cột vào ma trận.

Một phương pháp khác nữa để thay đổi ma trận trong R là thêm các hàng hoặc cột mới hoặc cả hai sử dụng hàm `rbind()` và `cbind()`. Trong ví dụ này, chúng ta tạo một ma trận mới đặt tên biến là `'new_mat'` với 3 dòng 3 cột.

```
> new_mat = matrix(1:12, nrow = 3, ncol = 3)
> new_mat
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Bây giờ chúng ta sẽ thêm một cột mới vào sử dụng hàm: `cbind()`.

```
> new_mat <- cbind(new_mat, c(1,2,3))
```

Hoặc chúng ta có thể thêm một dòng mới sử dụng hàm: `rbind()`.

```
> rbind(new_mat, c(1,2,3))
```

Lúc này kết quả trả về ta sẽ có một ma trận mới với 4 cột và 4 dòng. Tất nhiên là bạn phải lưu lại ma trận mới này, chứ hàm này không hỗ trợ việc cập nhật lại ma trận mà chỉ đơn thuần tạo ra ma trận mới dựa trên dữ liệu muốn thêm và dữ liệu ban đầu.

```
> new_mat
      [,1] [,2] [,3] [,4]
[1,]    1    4    7    1
[2,]    2    5    8    2
[3,]    3    6    9    3
[4,]    1    2    3    1
```

- d. Cách kiểm tra không gian của một ma trận: `dim(biến)`

```
> dim(new_mat)
[1] 4 4
```

Tức là ma trận của chúng ta có 4 dòng và 4 cột.

- e. Chúng ta có thể chuyển vị ma trận sử dụng hàm `t()`

```
> t(new_mat)
      [,1] [,2] [,3] [,4]
[1,]    1    2    3    1
[2,]    4    5    6    2
[3,]    7    8    9    3
[4,]    1    2    3    1
```

Các toán tử trên ma trận:

- a. Phép cộng (+).

Để biểu diễn phép cộng ma trận trong R, chúng ta tạo 2 ma trận có cùng số dòng và số cột như sau:

```
> mat1 <- matrix(data = 1:8, nrow = 4, ncol = 4)
> mat2 <- matrix(data = 1:16, nrow = 4, ncol = 4)
```

Chúng ta sẽ sử dụng hai ma trận này cho bốn phép toán bên dưới.

Để thực hiện phép cộng giữa hai ma trận, ta đơn giản sử dụng '+' như sau:

```
> sum <- mat1 + mat2
> sum
      [,1] [,2] [,3] [,4]
[1,]    2   10   10   18
[2,]    4   12   12   20
[3,]    6   14   14   22
[4,]    8   16   16   24
```

b. Phép trừ (-).

Để trừ hai ma trận, chúng ta sử dụng '-' như sau:

```
> sub <- mat1 - mat2
> sub
      [,1] [,2] [,3] [,4]
[1,]    0    0  -8  -8
[2,]    0    0  -8  -8
[3,]    0    0  -8  -8
[4,]    0    0  -8  -8
```

c. Nhân ma trận với một số.

Để nhân ma trận với một số, chúng ta chỉ việc lấy ma trận nhân với số đó sử dụng '*' như sau:

```
> prod <- mat1*4
> prod
      [,1] [,2] [,3] [,4]
[1,]    4   20    4   20
[2,]    8   24    8   24
[3,]   12   28   12   28
[4,]   16   32   16   32
```

d. Phép chia (/)

Để thực hiện phép chia hai ma trận, chúng ta sử dụng '/' như sau:

```
> div <- mat1/mat2
> div
      [,1] [,2] [,3] [,4]
[1,]    1    1 0.1111111 0.3846154
[2,]    1    1 0.2000000 0.4285714
[3,]    1    1 0.2727273 0.4666667
[4,]    1    1 0.3333333 0.5000000
```

e. Ma trận đơn vị.

Chúng ta có thể tạo được một ma trận đơn vị $n \times n$ sử dụng hàm `diag(n)`.

```
> diag(4)
      [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
```

f. Các ứng dụng của ma trận R:

- Trong hình học, ma trận dùng cho sự khảo sát các biến và bộ dataset, dùng cho vẽ đồ thị - plotting graphs, thống kê - statistics, và trong nhiều lĩnh vực khác.
- Trong robotics và tự động hoá, ma trận là thành phần quan trọng nhất để robot di chuyển.

- Ma trận được sử dụng trong tính tổng sản phẩm tiêu thụ trong nước đối với kinh tế. Do đó, nó giúp tính toán hiệu quả giữa mối quan hệ giữa hàng hoá và sản phẩm.
- Trong ứng dụng máy tính, ma trận có vai trò quan trọng trong việc tái thiết từ phép chiếu ảnh dạng 2 chiều sang cấu trúc 3 chiều và ngược lại tái cấu trúc dữ liệu 3 chiều lên màn hình 2 chiều, tạo ra các cảnh chuyển động như thực tế.
- Trong vật lý ứng dụng, ma trận có thể dùng để nghiên cứu các mạch điện.
- ...

4. Mảng trong R.

Trong ngôn ngữ lập trình R, mảng là cấu trúc dữ liệu đa chiều - Multi-dimensional. Thực tế thì dữ liệu thường được lưu dưới dạng ma trận, các hàng cũng như các cột. Chúng ta có thể dùng cấp của ma trận, chỉ số dòng và chỉ số cột để truy xuất thành phần của ma trận.

Mảng trong R có thể là các đối tượng chứa dữ liệu mà có thể lưu trữ nhiều hơn là hai chiều. Tạo mảng với hàm `array()`. Chúng ta có thể sử dụng vector chứa dữ liệu như là đầu vào. Để tạo được một mảng, chúng ta cần một tham số đầu vào là số chiều của mảng.

Cú pháp mảng trong R:

```
> Array_NAME <- array(data, dim = (row_Size, column_Size, matrices, dimnames))
```

Trong đó :

- **data**: dữ liệu đầu vào với dạng vector đã nói ở trên.
- **matrices**: số lượng ma trận muốn tạo ra từ data.
- **row_size**: số phần tử trên một dòng của mảng có thể chứa.
- **column_size**: số phần tử trên cùng một cột của mảng có thể chứa.
- **dimnames**: Có thể dùng để đổi tên hàng và cột của ma trận theo ý muốn của người dùng, nếu không truyền tham số vào thì mặc định sẽ để trống.

Ví dụ:

Trong ví dụ dưới đây, chúng ta sẽ tạo ra một mảng trong R gồm hai ma trận 3x3 với mỗi ma trận có 3 dòng và 3 cột.

```
> vec1 <- c(1,2,4,9,10,11,15,27)
> vec2 <- c(15,17,27,3,10,11)
> output <- array(c(vec1,vec2),dim = c(3,3,2))
> output
,, 1
      [,1] [,2] [,3]
[1,]    1    9   15
[2,]    2   10   17
[3,]    4   11   27

,, 2
      [,1] [,2] [,3]
[1,]    3    1    9
[2,]   10    2   10
[3,]   11    4   11
```

Chúng ta có thể thấy, đây là 2 ma trận khác nhau được từ 2 biến `vec1`, `vec2`, với các phần tử được nối với nhau liên tục theo thứ tự biến được truyền vào và lấp đầy 2 ma trận cho đến khi đủ.

Tham số `dim = c(3,3,2)` được sử dụng để tạo ra 2 ma trận 3x3 với 3 hàng và 3 cột, ngoài ra người đọc cũng có thể thử lại với số dòng, số cột, số lượng ma trận khác để hiểu rõ hơn cách vận hành của hàm `array()`.

Một số thao tác khác có thể thực hiện trên hàng và cột trong ma trận R:

- Đặt tên cho cột và hàng.

```
> column.names <- c("COL1","COL2","COL3")
> row.names <- c("ROW1","ROW2","ROW3")
```

```
> matrix.names <- c("Matrix1","Matrix2")
>result <- array(c(vec1,vec2),dim = c(3,3,2),
dimnames = list(row.names,column.names, matrix.names))
>result
,, Matrix1

      COL1 COL2 COL3
ROW1     1     9    15
ROW2     2    10    17
ROW3     4    11    27

,, Matrix2

      COL1 COL2 COL3
ROW1     3     1     9
ROW2    10     2    10
ROW3    11     4    11
```

- Truy cập vào phần tử trong mảng R.
In ra hàng thứ 3 của ma trận thứ 2.

```
> result[3,,2]
COL1 COL2 COL3
 11     4    11
```

In ra phần tử ở hàng thứ 1 và cột thứ 3 của ma trận thứ nhất.

```
> result[1,3,1]
[1] 15
```

In ra ma trận thứ 2.

```
> result[, ,2]
      COL1 COL2 COL3
ROW1     3     1     9
ROW2    10     2    10
ROW3    11     4    11
```

Vậy chúng ta có thể thấy rõ được thứ tự các giá trị được thể hiện bên trong mảng. `result[r,c,d]` trong đó `r` là hàng thứ `r`, `c` là cột thứ `c`, `d` là ma trận thứ `d` tồn tại trong mảng R.

- Tính toán trong mảng R.
Cú pháp :

```
> apply(x, margin, fun)
```

Trong đó :

- `x`: mảng cần tính toán.
- `margin`: phương thức tính : theo hàng `c(1)`, theo cột `c(2)`.
- `fun`: hàm để xử lý/tính toán các phần tử trong mảng.

Ví dụ: Trong ví dụ bên dưới, chúng ta sẽ tính tổng các phần tử theo cột trên cả 2 ma trận.

```
> vector1 <- c(1,2,3)
> vector2 <- c(3,4,5,6,7,8)
> new.array <- array(c(vector1,vector2),dim = c(3,3,2))
> result <- apply(new.array, c(2), sum)
> result
```

Thật ra, mảng là một cấu trúc dữ liệu rất mạnh của R, ứng dụng rõ nhất là trong việc biểu diễn biểu đồ, tuy nhiên chúng ta sẽ chưa đề cập tới trong phần này.

5. List trong R.

List là kiểu cấu trúc dữ liệu chứa nhiều phần tử là các kiểu dữ liệu khác nhau như là xâu, số, vector, hoặc là một **list** khác trong nó. List cũng có thể chứa một ma trận hoặc là một hàm như là một phần tử trong nó. Nói cách khác, **list** khá giống với kiểu dữ liệu chúng ta đã nói đến **vector** ở trên khi chúng có thể chứa các kiểu dữ liệu khác.

Ví dụ:

```
> num_list = c(3,4,5)
> char_list = c("a", "b", "c", "d", "e")
> logic_list = c(TRUE, TRUE, FALSE, TRUE)
> out_list = list(num_list, char_list, logic_list, 3)
> out_list
[[1]]
[1] 3 4 5

[[2]]
[1] "a" "b" "c" "d" "e"

[[3]]
[1] TRUE TRUE FALSE TRUE

[[4]]
[1] 3
```

6. Data Frame trong R.

Data Frame là cấu trúc dữ liệu dạng bảng: đại diện cho các biến hay là mẫu - sample, mỗi mẫu sẽ bao gồm nhiều quan sát hoặc là đo lường trên mỗi cột.

Data Frame được dùng như một bảng dữ liệu, nó là list của các vector có cùng độ dài.

Ví dụ:

```
> num_list = c(3,4,5)
> char_list = c("a", "b", "c")
> logic_list = c(TRUE, FALSE, TRUE)
> data_frame = data.frame(num_list, char_list, logic_list)
> data_frame
  num_list char_list logic_list
1        3         a        TRUE
2        4         b       FALSE
3        5         c        TRUE
```

Data Frame giống như list hay mảng, nó có thể lưu trữ dữ liệu dưới dạng cột mang nhiều kiểu dữ liệu khác nhau.

Các đặc điểm của Data Frame:

- Tên cột luôn không được để trống.
- Mỗi tên hàng là duy nhất trong mỗi bộ dữ liệu khác nhau.
- Dữ liệu trong Data Frame được trình bày dưới dạng bảng, có thể chứa số thực, hệ số, kí tự.
- Mỗi phần tử trong cùng một cột phải cùng kiểu dữ liệu.

Tóm lại, chúng ta đã nói qua cấu trúc dữ liệu trong R và các cách dùng và giải thích, ví dụ về nó[2]. Chúng ta sẽ tiếp tục tìm hiểu kỹ hơn trong các phần tiếp theo. Chúng tôi hi vọng bạn đọc thích đề tài này của chúng tôi.

Tài liệu

1. John Chambers. *Programming with Data (The Green Book)*. Springer, Jan 1, 2001.
2. DataFlair. Data structures in r - the most essential concept for r aspirants!
3. Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics, journal of computational and graphical statistics, volume 5, number 3, pages 299-314. 1996.
4. Roger D. Peng. R programming for data science.
5. www.psychologicalscience.org. Why you-should become a user a brief introduction to r.