

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÁO CÁO TIẾN TRÌNH THỰC HÀNH
MÔN CS313.M21: KHAI THÁC DỮ LIỆU VÀ ỨNG DỤNG
CHỦ ĐỀ: NGÔN NGỮ LẬP TRÌNH R

Giảng viên hướng dẫn: ThS. Nguyễn Thị Anh Thư

Sinh viên thực hiện:

Họ và tên	MSSV
1. Lê Võ Ngọc Anh	18520452
2. Nguyễn Hữu Trường	18521564
3. Nông Thanh Hồng	19521551
4. Trương Thế Tấn	19522180
5. Nguyễn Thị Hiền Trang	19522383

TP. Hồ Chí Minh, tháng 3, năm 2022

MỤC LỤC

I.	Tổng quan về ngôn ngữ lập trình R	1
1.	Giới thiệu về R	1
2.	Môi trường R	1
3.	Lịch sử hình thành R	2
4.	Cấu trúc dự án mã nguồn mở R.....	2
5.	Gói (package) trong R	2
II.	Độ phổ biến và Ứng dụng của R	7
1.	Mức độ phổ biến của R	7
2.	Ứng dụng thực tế của R.....	7
3.	Ngôn ngữ lập trình R được ứng dụng trong các ngành	8
III.	Có nên chọn và sử dụng R	11
1.	Tại sao nên chọn R	11
2.	Tại sao nên học R	13
IV.	Ưu điểm, hạn chế của R và So sánh R với Python.....	14
1.	Ưu và nhược điểm của R.....	14
2.	So sánh R với Python	15
V.	Hướng dẫn sử dụng R cơ bản	18
1.	Giao tiếp với R	18
2.	Cài đặt R trong window.....	20
3.	Cài đặt RStudio	20
4.	Các thư viện phổ biến trong R.....	22
5.	Cú pháp cơ bản của R.....	33
6.	Tutorial R.....	37
6.1.	Toán tử R	37
6.2.	Cấu trúc dữ liệu R.....	38
6.3.	Cấu trúc điều khiển.....	48
6.5.	Đồ họa cơ sở R	49
VI.	Demo chương trình.....	65
VII.	Kết luận	69
	TÀI LIỆU THAM KHẢO	70

DANH MỤC HÌNH ẢNH

Hình 1. 1. Số lượng các gói hỗ trợ R qua các năm trên CRAN	3
Hình 1. 2. Các gói trong R được hỏi - trả lời nhiều nhất trên Stack Overflow	3
Hình 1. 3. Cài đặt các gói thông qua giao diện trên RStudio	4
Hình 1. 4. Ví dụ biểu đồ vẽ bằng gói ggplot2 trên R	5
Hình 2. 1. Xếp hạng mức độ phổ biến của R trong 2019-2020.....	7
Hình 2. 2. Mức độ phổ biến của R trong các ngành công nghiệp	9
Hình 2. 3. Các công ty trên thế giới sử dụng R	11
Hình 3. 1. Mức độ thích hợp của các công cụ phân tích dữ liệu hiện nay trong doanh nghiệp	12
Hình 3. 2. Mức độ phổ biến của một số ngôn ngữ lập trình trên Stack Overflow	13
Hình 5. 1. Giao diện của RStudio.....	18
Hình 5. 2. Giao diện web cơ bản của gói shiny	18
Hình 5. 3. Các tùy chọn tạo Document trong R	19
Hình 5. 4. Các tùy chọn tạo bản Presentation trong R	19
Hình 5. 5. Cài đặt R bước 1: chọn hệ điều hành.....	20
Hình 5. 6. Cài đặt R bước 2: chọn tùy chọn base	20
Hình 5. 7. Cài đặt R bước 3: chọn phiên bản mới nhất và tải về	20
Hình 5. 8. Cài đặt RStudio bước 1: Các tùy chọn phiên bản của RStudio.....	21
Hình 5. 9. Cài đặt RStudio bước 2: lựa chọn phiên bản phù hợp với HĐH và tải về ..	21
Hình 5. 10. Cài đặt RStudio: Giao diện với theme tối của RStudio.....	21
Hình 5. 11. Mô hình của thuật toán SVM	26
Hình 5. 12. Mô hình thuật toán KNN.....	30
Hình 5. 13. Mô hình biểu đồ thanh ngang - Bar plot trong R	49
Hình 5. 14. Biểu đồ hộp - Box whisker trong R.....	54
Hình 5. 15. Biểu đồ Histogram trong R.....	57
Hình 5. 16. Biểu đồ tròn - Pie trong R.....	60
Hình 5. 17. Biểu đồ Quantile-Quantile (QQ) trong R	62

Hình 6. 1. Các lớp trong tập dữ liệu Iris.....	65
Hình 6. 2. So sánh kết quả của các thuật toán SVM, KNN, RandomForest, NavieBayes	66
Hình 6. 3. Kết quả dự đoán của mô hình KNN trên tập validation	66
Hình 6. 4. Kết quả mô hình XgBoost trên tập validation	68

I. Tổng quan về ngôn ngữ lập trình R

1. Giới thiệu về R

R là ngôn ngữ lập trình và môi trường cho tính toán thống kê và đồ họa mà các chuyên gia dữ liệu trên toàn thế giới sử dụng cho mọi thứ, từ lập bản đồ các xu hướng tiếp thị và xã hội rộng rãi trực tuyến đến phát triển các mô hình tài chính và khí hậu giúp thúc đẩy nền kinh tế và cộng đồng của chúng ta.

R cung cấp nhiều loại thống kê (mô hình tuyến tính và phi tuyến, kiểm tra thống kê cổ điển, phân tích chuỗi thời gian, phân loại, phân cụm,...) và các kỹ thuật đồ họa, và có khả năng mở rộng cao.

R có sẵn dưới dạng Phần mềm Miễn phí theo các điều khoản của Giấy phép Công cộng GNU (*GNU General Public License*) của Tổ chức Phần mềm Tự do (Free Software Foundation) ở dạng mã nguồn. R biên dịch và chạy trên nhiều nền tảng UNIX và các hệ thống tương tự (bao gồm FreeBSD và Linux), Windows và MacOS.

2. Môi trường R

R là một bộ phần mềm tích hợp để thao tác dữ liệu, tính toán và hiển thị đồ họa. R bao gồm:

- Một phương tiện lưu trữ và xử lý dữ liệu hiệu quả.
- Một bộ các toán tử để tính toán trên mảng, đặc biệt là các ma trận.
- Một bộ sưu tập lớn, chặt chẽ, tích hợp các công cụ trung gian để phân tích dữ liệu.
- Các phương tiện đồ họa để phân tích dữ liệu và hiển thị trên màn hình hoặc trên bản cứng.
- Một ngôn ngữ lập trình được phát triển tốt, đơn giản và hiệu quả bao gồm các điều kiện, vòng lặp, các hàm đệ quy do người dùng định nghĩa và các phương tiện đầu vào và đầu ra.

Thuật ngữ “môi trường” nhằm mô tả nó như một hệ thống được hoạch định đầy đủ và chặt chẽ, chứ không phải là sự tích lũy gia tăng của các công cụ rất cụ thể và không linh hoạt, như trường hợp thường xảy ra với các phần mềm phân tích dữ liệu khác.

R được thiết kế dựa trên một ngôn ngữ máy tính thực sự và nó cho phép người dùng thêm chức năng bổ sung bằng cách xác định các chức năng mới. Đối với các tác vụ đòi hỏi nhiều tính toán, mã C, C++ và Fortran có thể được liên kết và gọi trong thời gian chạy. Người dùng nâng cao có thể viết mã C để thao tác trực tiếp các đối tượng R.

3. Lịch sử hình thành R

R được thực hiện lần đầu tiên vào đầu những năm 1990 bởi Robert Gentleman và Ross Ihaka, cả hai đều là giảng viên tại Đại học Auckland. Ngôn ngữ R được mô phỏng chặt chẽ dựa trên Ngôn ngữ S dành cho tính toán thống kê do John Chambers, Rick Becker, Trevor Hastie, Allan Wilks và những người khác tại Bell Labs hình thành vào giữa những năm 1970 và được công bố rộng rãi vào đầu những năm 1980. Robert và Ross đã thành lập R như một dự án mã nguồn mở vào năm 1995, kể từ năm 1997, dự án R được quản lý bởi R Core Group. Và, vào tháng 2 năm 2000, bản phát hành đầu tiên của R.

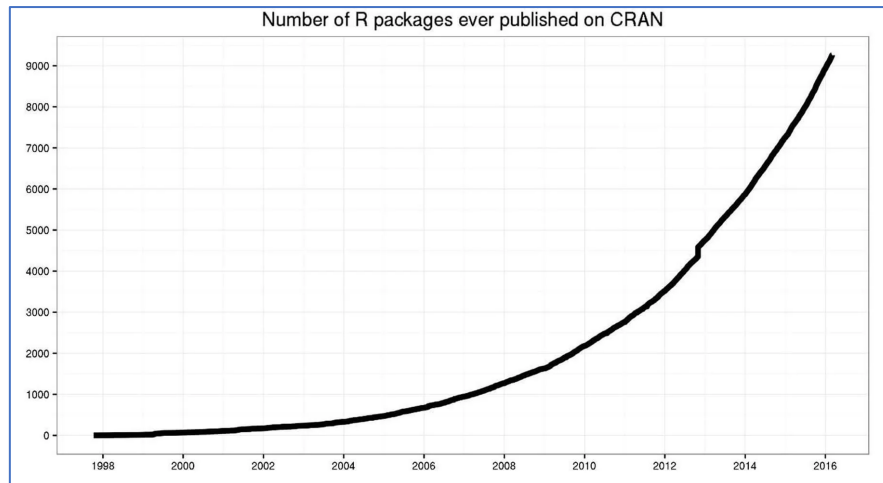
Họ cùng với nhiều người khác tiếp tục làm việc và sử dụng R. Họ tiếp tục tạo ra các công cụ mới cho R và tạo ra các ứng dụng mới cho R mỗi ngày. Có hơn 10.000 thư viện do người dùng tạo được xây dựng để nâng cao chức năng của R. Các gói này có sự hỗ trợ và xác nhận chất lượng từ nguồn cộng đồng từ các nhà lãnh đạo được công nhận trong mọi lĩnh vực.

4. Cấu trúc dự án mã nguồn mở R

Trung tâm của Dự án nguồn mở R và Cộng đồng R (*R Open Source Project and R Community*) là **R Core**, một nhóm khoảng 20 nhà phát triển duy trì R và hướng dẫn sự phát triển của nó. Cơ cấu công khai chính thức cho Cộng đồng R được cung cấp bởi **R Foundation**, một tổ chức phi lợi nhuận với danh sách ấn tượng các thành viên và những người ủng hộ. R Foundation đảm bảo sự ổn định tài chính của dự án R và nắm giữ và quản lý bản quyền của phần mềm R và tài liệu của nó.

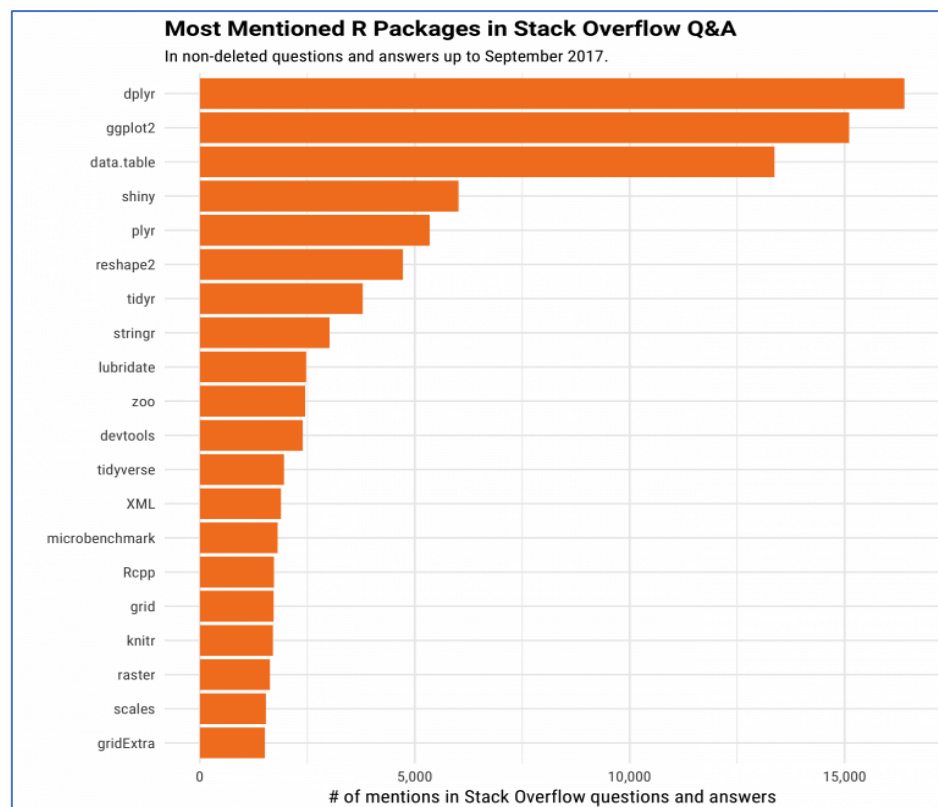
5. Gói (package) trong R

Tất cả các thư viện của R, gần 12.000 gói, được lưu trữ trong CRAN. CRAN là một mã nguồn mở và miễn phí. Bạn có thể tải xuống và sử dụng nhiều thư viện để thực hiện Học máy hoặc phân tích chuỗi thời gian.



Hình 1. 1. Số lượng các gói hỗ trợ R qua các năm trên CRAN

Các công dụng chính của R là và sẽ luôn là thống kê, trực quan hóa và học máy. Hình bên dưới cho thấy gói R nào có nhiều câu hỏi nhất trong Stack Overflow. Trong top 10, hầu hết chúng đều liên quan đến quy trình làm việc của một nhà khoa học dữ liệu: chuẩn bị dữ liệu và truyền đạt kết quả.



Hình 1. 2. Các gói trong R được hỏi - trả lời nhiều nhất trên Stack Overflow

- Cài đặt package trong R

Có hai cách đơn giản để cài đặt gói R bằng RStudio:

- Cách 1: là thực thi dòng mã sau trong bảng điều khiển

Cài đặt một package:

```
install.packages("name-package")
```

Cài đặt từ hai package trở lên:

```
install.packages(c("package-1", "package-2", ...))
```

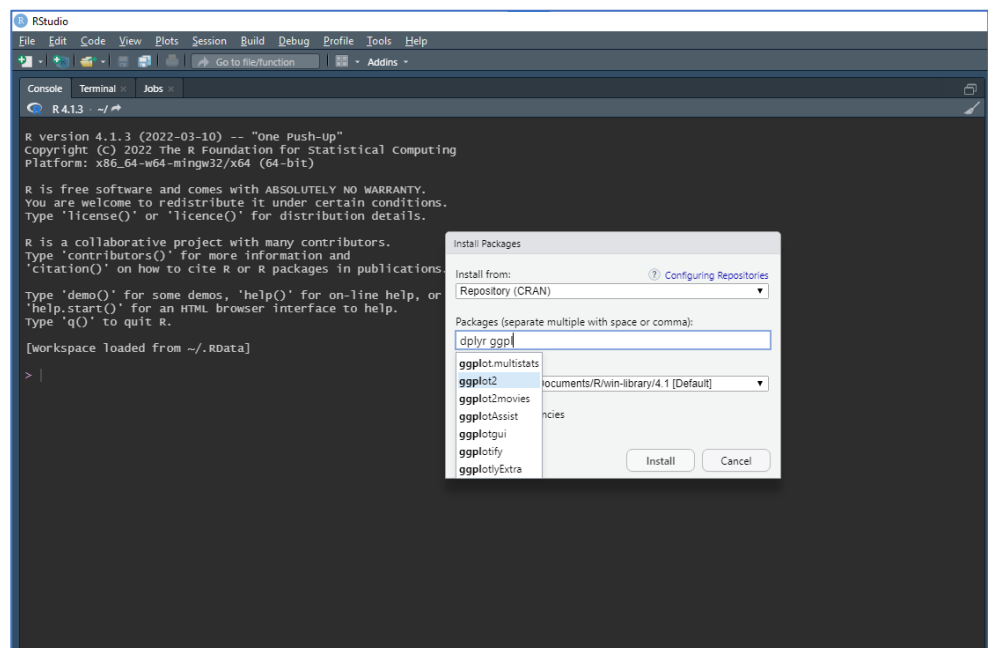
Ví dụ:

```
install.packages(c("dplyr", "ggplot2"))
```

```
install.packages("ggplot2")
```

- Cách 2: là một giao diện đồ họa để sử dụng được tích hợp sẵn trong RStudio mà từ đó bạn có thể tìm kiếm và tải xuống bất kỳ gói R nào có sẵn trên CRAN.

Trên thanh Menu chọn: *Tools* → *Install Packages...* Và nhập package muốn cài đặt. Sử dụng dấu cách (khoảng trắng) để cài đặt nhiều gói khác nhau.



Hình 1. 3. Cài đặt các gói thông qua giao diện trên RStudio

- Sử dụng package:

Để sử dụng một package trong mã, ta cần tải gói đó lên trong mã bằng câu lệnh:

```
library(package-name)
```

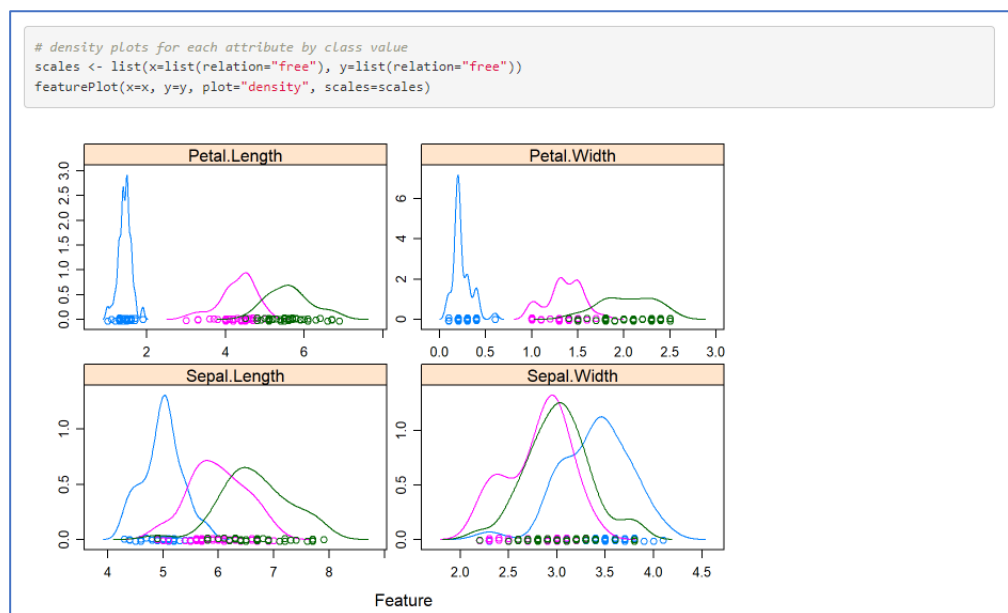
- Các packages phổ biến trong R

1. **dplyr** : `install.packages("dplyr")`

Đây là một trong những gói R được sử dụng nhiều nhất cho các tác vụ khoa học dữ liệu và học máy. **Dplyr** được sử dụng để thao tác dữ liệu bằng cách cung cấp các chức năng khác nhau như: `mutate()`, `select()`, `filter()`, `summarise()`, `arrange()`.

2. **ggplot2** : `install.packages("ggplot2")`

Được sử dụng rộng rãi nhất để trực quan hóa dữ liệu và phân tích dữ liệu khám phá.



Hình 1. 4. Ví dụ biểu đồ vẽ bằng gói ggplot2 trên R

3. **kernLab** : `install.packages("kernlab")`

Gói **kernLab** (*Kernel-Based Machine Learning Lab*) là một gói để phân loại, hồi quy, phân cụm, phát hiện tính mới, hồi quy lượng tử và giảm kích thước. Trong số các phương pháp khác 'kernlab' bao gồm Support Vector Machines (SVM), Spectral Clustering (Phân cụm quang phổ), Kernel PCA, Gaussian Processes và QP solver.

4. **caret** : `install.packages("caret")`

Gói **caret** (*Classification and Regression Training*) chứa một tập hợp các chức năng được sử dụng để tạo các mô hình dự đoán. Nó có các chức năng khác như lựa chọn tính năng (*feature selection*), tách dữ liệu, tiền xử

lý dữ liệu, điều chỉnh mô hình (*model tuning*), tầm quan trọng của tính năng và nhiều chức năng khác.

5. randomForest : `install.packages("randomForest")`

Gói randomForest được sử dụng để tạo các rừng ngẫu nhiên (*random forests*) bằng ngôn ngữ thống kê R. Gói này triển khai thuật toán rừng ngẫu nhiên của Breiman để phân loại và hồi quy. Nó cũng có thể được sử dụng ở chế độ không giám sát để đánh giá mức độ gần nhau giữa các điểm dữ liệu.

6. plotly : `install.packages("plotly")`

plotly là một gói R để tạo đồ thị dựa trên web tương tác thông qua thư viện vẽ đồ thị JavaScript nguồn mở plotly.js. Sử dụng gói này, bạn có thể tạo đồ họa web tương tác từ đồ thị 'ggplot2' hoặc giao diện tùy chỉnh cho thư viện JavaScript 'plotly.js' lấy cảm hứng từ ngữ pháp đồ họa.

Ví dụ chạy đoạn mã dưới đây trong RStudio:

7. superml : `install.packages("superml")`

Superml là một trong những gói R phổ biến dành cho máy học, cung cấp giao diện chuẩn cho người dùng sử dụng cả ngôn ngữ lập trình Python và R để xây dựng mô hình học máy. Gói này về cơ bản cung cấp các tính năng của Scikit Learn và dự đoán giao diện để đào tạo các mô hình học máy trong R.

8. dataExplorer : `install.packages("DataExplorer")`

DataExplorer là một trong những gói máy học phổ biến của R tập trung vào ba mục tiêu chính, đó là phân tích dữ liệu khám phá (*Exploratory Data Analysis - EDA*), kỹ thuật tính năng (*Feature Engineering*) và báo cáo dữ liệu (*Data Reporting*). Gói này tự động hóa quy trình thăm dò dữ liệu cho các nhiệm vụ phân tích và mô hình dự đoán để người dùng có thể tập trung vào việc hiểu dữ liệu và trích xuất thông tin chi tiết. Gói quét và phân tích từng biến, và hiển thị chúng bằng các kỹ thuật đồ họa điển hình.

9. knitr : `install.packages("knitr")`

knitr là một gói R để tạo báo cáo động có thể được sử dụng để tích hợp nhiều loại mã khác nhau vào mã R chẳng hạn như Markdown, LyX, LaTeX,

AsciiDoc, HTML, v.v ... thì Knir là một gói rất quan trọng cần có nếu bạn đang làm việc nghiên cứu để tạo báo cáo và nó cũng hỗ trợ rất nhiều trong việc tự động hóa quy trình dữ liệu từ phân tích dữ liệu đến tạo báo cáo về nó.

10.mlr3 : `install.packages("mlr3")`

Bạn có thể triển khai các mô hình học máy được giám sát (*Supervised learning*) và không giám sát (*Unsupervised Machine learning*) khác nhau trên Scikit-learning như Classification, Regression, Support Vector Machines, Random Forests, Nearest Neighbors, Naive Bayes, Decision Trees, Clustering,... với mlr3

II. Độ phổ biến và Ứng dụng của R

1. Mức độ phổ biến của R

Từ năm 2019 đến 2020, Ngôn ngữ lập trình R đã thiết lập một kỷ lục mới từ hạng 20 lên hạng 8. Mức độ phổ biến của R vẫn đang tăng lên trong dòng chảy của Python. Không thể phủ nhận được sự phổ biến, tiện dụng của R trong công cuộc lập trình thống kê dữ liệu.

Jul 2020	Jul 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	16.45%	+2.24%
2	1	▼	Java	15.10%	+0.04%
3	3		Python	9.09%	-0.17%
4	4		C++	6.21%	-0.49%
5	5		C#	5.25%	+0.88%
6	6		Visual Basic	5.23%	+1.03%
7	7		JavaScript	2.48%	+0.18%
8	20	▲▲	R	2.41%	+1.57%
9	8	▼	PHP	1.90%	-0.27%
10	13	▲	Swift	1.43%	+0.31%
11	9	▼	SQL	1.40%	-0.58%
12	16	▲▲	Go	1.21%	+0.19%
13	12	▼	Assembly language	0.94%	-0.45%
14	19	▲▲	Perl	0.87%	-0.04%
15	14	▼	MATLAB	0.84%	-0.24%

Hình 2. 1. Xếp hạng mức độ phổ biến của R trong 2019-2020

2. Ứng dụng thực tế của R

- Data Science

Trong thời đại IoT, các thiết bị tạo ra hàng terabyte dữ liệu có thể sử dụng để hỗ trợ ra quyết định, khoa học dữ liệu không có cách nào khác là

phải tiến lên. Ngôn ngữ R cho các nhà khoa học dữ liệu một công cụ mạnh mẽ để thu thập dữ liệu thời gian thực, đồng thời thực hiện phân tích thống kê và dự báo, tạo ra các kết quả trực quan dễ hiểu.

- **Tính toán thống kê**

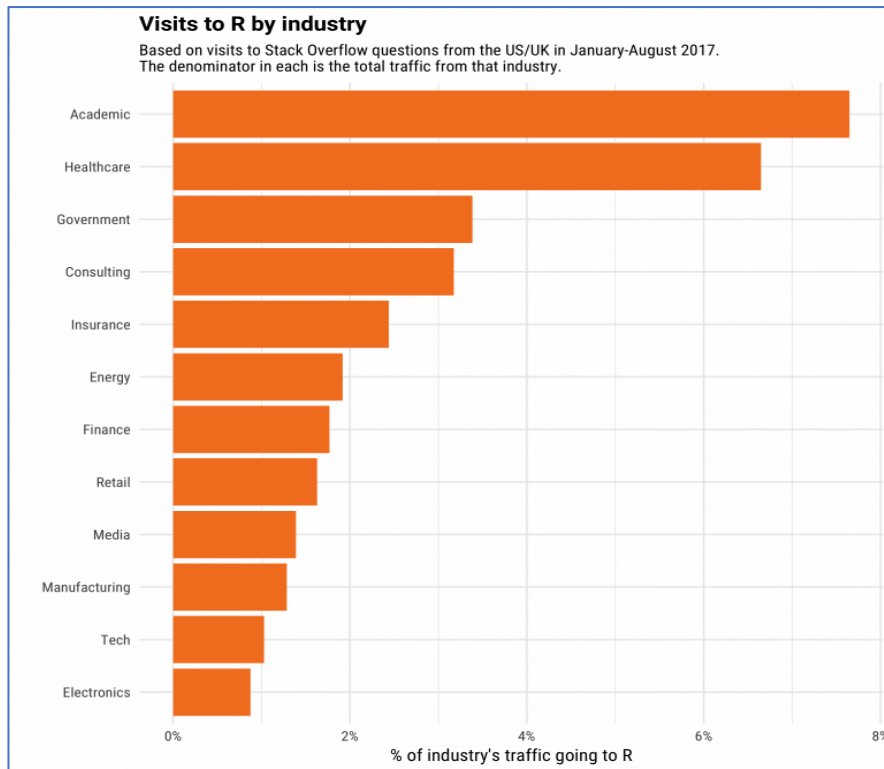
R là ngôn ngữ phổ biến nhất trong cộng đồng các nhà thống kê, với một kho package khổng lồ chứa gần 12.000 package đáp ứng mọi phép tính thống kê bạn có thể nghĩ ra. Cú pháp đặc biệt của R cho phép các nhà thống kê, kể cả những người không có nền tảng khoa học máy tính, cũng có thể nhanh chóng import, làm sạch và phân tích dữ liệu từ nhiều nguồn khác nhau, vẽ biểu đồ từ bất cứ dataset nào.

- **Học máy (Machine Learning)**

R cũng có nhiều package thực hiện các thao tác machine learning như hồi quy tuyến tính, phi tuyến tính, cây quyết định, ... R còn được sử dụng để cài đặt các thuật toán trong lĩnh vực tài chính, nghiên cứu di truyền, marketing hay chăm sóc sức khỏe.

3. Ngôn ngữ lập trình R được ứng dụng trong các ngành

Nếu chúng ta chia nhỏ việc sử dụng R theo ngành, chúng ta thấy rằng giới học thuật được ưu tiên hàng đầu. R là một ngôn ngữ để làm thống kê. R là lựa chọn đầu tiên trong ngành chăm sóc sức khỏe, tiếp theo là chính phủ và tư vấn.



Hình 2. 2. Mức độ phổ biến của R trong các ngành công nghiệp

Các trường hợp sử dụng thực tế của ngôn ngữ lập trình R trong các cơ quan/ tổ chức theo ngành khác nhau:

- *R sử dụng trong nghiên cứu & học thuật*
 - **Đại học Cornell** : Cornell khuyến nghị các nhà nghiên cứu và sinh viên của họ sử dụng R cho tất cả các nghiên cứu của họ liên quan đến tính toán thống kê.
 - **UCLA** : Đại học California, Los Angeles sử dụng R để giảng dạy thống kê và phân tích dữ liệu cho sinh viên của mình.
- *R Các trường hợp sử dụng trong lĩnh vực công nghệ thông tin*
 - **Mozilla** : Mozilla sử dụng R để trực quan hóa hoạt động web cho trình duyệt firefox của họ.
 - **Microsoft** : Microsoft sử dụng R làm công cụ thống kê trong khuôn khổ Azure Machine Learning. Họ cũng sử dụng nó cho dịch vụ mai mối Xbox.
 - **Foursquare** : R hoạt động ở chế độ hậu trường trên công cụ đề xuất của Foursquare.

- **Google** : Google sử dụng R để cải thiện kết quả tìm kiếm, cung cấp các đề xuất tìm kiếm tốt hơn, để tính toán ROI của các chiến dịch quảng cáo của họ, để tăng hiệu quả của quảng cáo trực tuyến và dự đoán hoạt động kinh tế của họ.
- *R sử dụng trong tài chính*
 - **Lloyds of London** : Lloyds of London sử dụng R để phân tích rủi ro.
 - **Bảo hiểm Bajaj Allianz** : Bajaj Allianz sử dụng R để tạo ra các mô hình xu hướng bán thêm và công cụ khuyến nghị của họ. Họ cũng sử dụng nó để khai thác dữ liệu và tạo ra thông tin chi tiết hữu ích để cải thiện trải nghiệm của khách hàng.
- *R sử dụng trong thương mại điện tử*
 - **Amazon** : Amazon sử dụng R và phân tích dữ liệu để cải thiện các đề xuất sản phẩm chéo của họ.
 - **Flipkart** : Flipkart sử dụng R để phân tích dự đoán giúp họ đưa ra các quảng cáo được nhắm mục tiêu.
- *R sử dụng trong phương tiện truyền thông xã hội*
 - **Facebook**: Facebook sử dụng R để dự đoán các tương tác của đồng nghiệp và cập nhật biểu đồ mạng xã hội của mình.
 - **Twitter**: Twitter sử dụng R để phân cụm ngữ nghĩa. Họ cũng sử dụng nó để trực quan hóa dữ liệu.
- *R sử dụng trong ngân hàng*
 - **ANZ** : Ngân hàng ANZ sử dụng R cho mô hình rủi ro tín dụng và cả trong các mô hình cho tổn thất thế chấp.
 - **Bank of America** : Bank of America sử dụng R để báo cáo tài chính và tính toán thiệt hại tài chính
- *R sử dụng trong chăm sóc sức khỏe*
 - **Merck** : Merck & co. sử dụng ngôn ngữ lập trình R cho các thử nghiệm lâm sàng và thử nghiệm thuốc
- *R sử dụng trong sản xuất*

- **Ford Motor Company** : Ford sử dụng R để phân tích thống kê nhằm hỗ trợ chiến lược kinh doanh của mình và phân tích tình cảm của khách hàng về sản phẩm của mình, giúp họ cải thiện thiết kế trong tương lai.
- **John Deere** : John Deere sử dụng R để dự báo nhu cầu về sản phẩm và phụ tùng thay thế của họ. Họ cũng sử dụng nó để dự báo năng suất cây trồng và sử dụng dữ liệu đó cho chiến lược kinh doanh của họ và để đáp ứng nhu cầu thị trường cũng như suy thoái.
- *R sử dụng trong các hoạt động của chính phủ*
 - **Cơ quan Quản lý Thực phẩm và Dược phẩm**: FDA sử dụng R để đánh giá thuốc và thực hiện các thử nghiệm tiền lâm sàng. Nó cũng sử dụng R để dự đoán các phản ứng có thể xảy ra và các vấn đề y tế do các sản phẩm thực phẩm khác nhau gây ra.
 - **Dịch vụ thời tiết quốc gia**: Dịch vụ thời tiết quốc gia sử dụng R để dự báo thời tiết và dự báo thiên tai. Họ cũng sử dụng nó để trực quan hóa các dự báo và dự đoán của họ để phân tích các khu vực bị ảnh hưởng.

Các công ty sử dụng R cho các vấn đề của doanh nghiệp



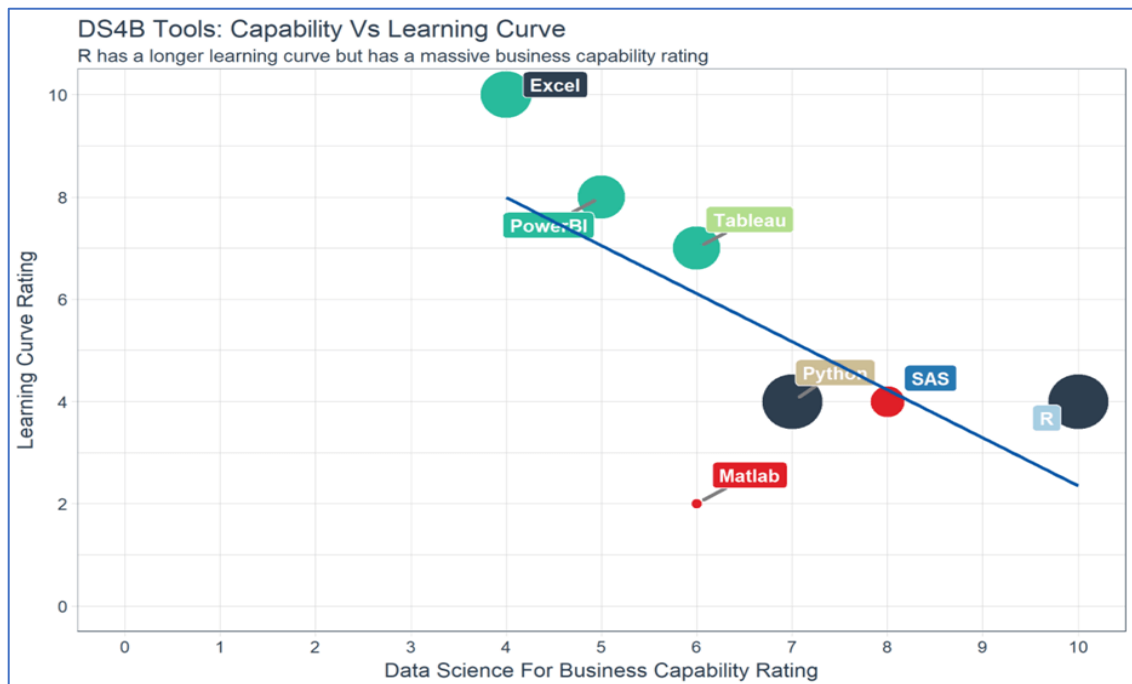
Hình 2. 3. Các công ty trên thế giới sử dụng R

III. Có nên chọn và sử dụng R

1. Tại sao nên chọn R

Khoa học dữ liệu đang định hình cách các công ty điều hành doanh nghiệp của họ. Câu hỏi lớn là bạn nên sử dụng công cụ / ngôn ngữ nào? Hiện nay có rất nhiều công cụ có sẵn trên thị trường để thực hiện phân tích dữ liệu. Hình ảnh dưới đây mô tả đường cong học tập so với khả năng kinh doanh mà một ngôn ngữ mang

lại. Nếu bạn muốn đưa ra cái nhìn sâu sắc nhất từ dữ liệu, thì bạn cần dành thời gian tìm hiểu công cụ thích hợp, đó là R.

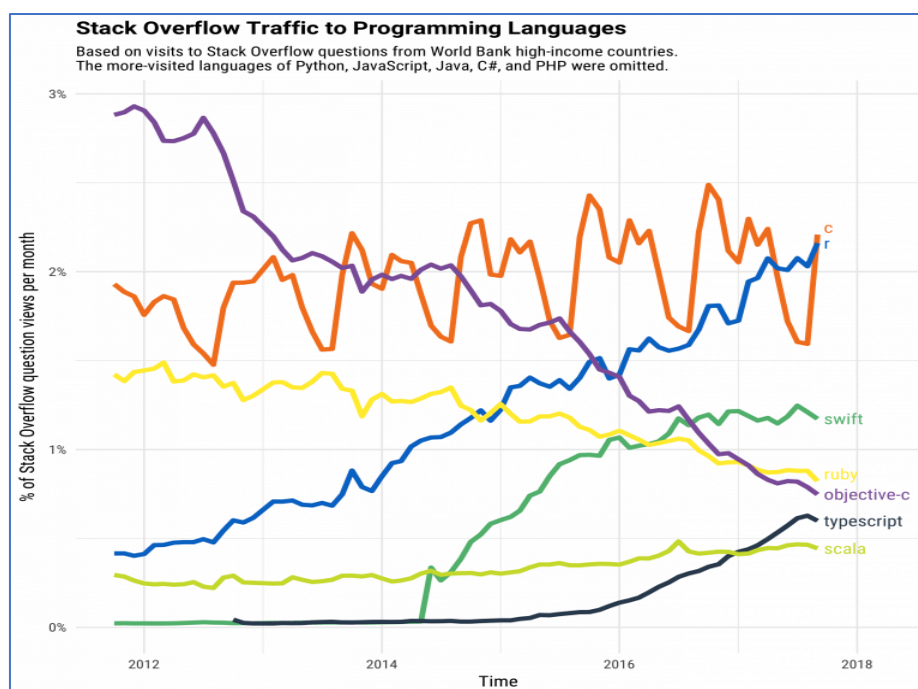


Hình 3. 1. Mức độ thích hợp của các công cụ phân tích dữ liệu hiện nay trong doanh nghiệp

Ở trên cùng bên trái của biểu đồ, bạn có thể thấy Excel và PowerBI. Hai công cụ này rất đơn giản để học nhưng không mang lại khả năng kinh doanh vượt trội, đặc biệt là về mặt mô hình hóa. Ở giữa, bạn có thể thấy Python và SAS. SAS là một công cụ chuyên dụng để chạy phân tích thống kê cho doanh nghiệp, nhưng nó không miễn phí. SAS là một phần mềm nhấp và chạy. Tuy nhiên, Python là một ngôn ngữ có đường cong học tập đơn điệu. Python là một công cụ tuyệt vời để triển khai Học máy và AI nhưng thiếu các tính năng giao tiếp. Với một đường cong học tập giống hệt nhau, R là sự cân bằng tốt giữa việc triển khai và phân tích dữ liệu.

Khi nói đến trực quan hóa dữ liệu (DataViz), bạn có thể đã nghe nói về Tableau. Không nghi ngờ gì nữa, Tableau là một công cụ tuyệt vời để khám phá các mẫu thông qua đồ thị và biểu đồ. Bên cạnh đó, việc học Tableau không tốn nhiều thời gian. Một vấn đề lớn với trực quan hóa dữ liệu là bạn có thể không bao giờ tìm thấy một mẫu hoặc chỉ tạo ra nhiều biểu đồ vô dụng. Tableau là một công cụ tốt để hình dung nhanh dữ liệu hoặc Business Intelligence. Khi nói đến thống kê và công cụ ra quyết định, R thích hợp hơn.

Stack Overflow là một cộng đồng lớn dành cho các ngôn ngữ lập trình. Nếu bạn gặp vấn đề về mã hóa hoặc cần hiểu một mô hình, Stack Overflow luôn sẵn sàng trợ giúp. Trong năm qua, phần trăm lượt xem câu hỏi đối với R đã tăng mạnh so với các ngôn ngữ khác. Xu hướng này tất nhiên có mối tương quan cao với thời đại bùng nổ của khoa học dữ liệu, nhưng nó phản ánh nhu cầu của ngôn ngữ R đối với khoa học dữ liệu.



Hình 3. 2. Mức độ phổ biến của một số ngôn ngữ lập trình trên Stack Overflow

2. Tại sao nên học R

Những nhà tuyển dụng xem R là một kỹ năng hữu ích và có giá trị. Điều này đặc biệt đúng trong bất kỳ ngành nào dựa vào phân tích dữ liệu.

Lý do các công ty/ tổ chức sử dụng R và tuyển dụng những lập trình viên có kinh nghiệm trong việc sử dụng R trong khoa học dữ liệu là:

- **Thu thập, phân tích dữ liệu dễ dàng**
 - Bằng cách sử dụng R, bạn có thể thực hiện thu thập dữ liệu, làm sạch và phân tích tất cả ở một nơi.
- **R rất mạnh cho việc phân tích dữ liệu**
 - Bạn có thể chạy code mà không cần đến bất cứ compiler nào - R là ngôn ngữ thông dịch (interpreted language). Do đó code có thể chạy mà

không cần compiler. R thông dịch code và làm cho việc viết code đơn giản hơn, dễ phát triển hơn.

- *Bất kỳ một phép tính nào cũng có thể thực hiện trên vectors* - R là một ngôn ngữ vector, do đó chúng ta có thể dùng bất kỳ hàm nào trên một vector mà không cần phải dùng vòng lặp.

Ví dụ: bạn có một mảng và phải tăng mỗi phần tử lên +1. Nếu không sử dụng vector, bạn sẽ cần lặp qua tất cả phần tử và cần n phép +1 cho n phần tử. Nếu bạn lưu mảng đó vào vector thì chỉ cần 1 phép +1 là xong.

- *Đây là Statistical-Language* - R được dùng trong sinh học, di truyền học và thống kê dữ liệu. R là ngôn ngữ turing-complete có nghĩa nó có thể hoàn thành bất kỳ thuật toán nào.

- **R được dùng nhiều trong kinh doanh**

- R là một open-source, nên nó cực kỳ "kinh tế". Đồng thời R rất phù hợp cho việc mô phỏng dữ liệu qua bảng biểu. Nhờ một cộng đồng phát triển và hơn 12000+ packages trong mọi lĩnh vực nghiên cứu. Hiện tại, khó có một công cụ nào có thể theo kịp R.
- Trong việc nghiên cứu dữ liệu, việc khan hiếm nhân tài là một vấn đề rất lớn. Các công ty có thể dùng ngôn ngữ R để làm nền tảng và training nhân viên sử dụng nó.

IV. Ưu điểm, hạn chế của R và So sánh R với Python

1. Ưu và nhược điểm của R

- *Ưu điểm của ngôn ngữ R*

- R có những package thống kê toàn diện nhất với công nghệ mới nhất, những ý tưởng mới thường xuất hiện đầu tiên trên R.
- R là open-source nên bất kỳ ai cũng có thể sử dụng và cải tiến nó.
- Vì là open-source nên R có thể được dùng mọi lúc mọi nơi cho bất cứ việc gì, kể cả bán các sản phẩm từ R theo điều kiện của giấy phép.
- R có thể chạy trên bất kỳ hệ điều hành nào.
- Bất kỳ ai cũng được hỗ trợ để đưa ra ý tưởng phát triển, fix bug, phát triển package mới.

- *Hạn chế của ngôn ngữ R*

- Một vài package của R có thể không hoàn hảo và còn lỗi
- Không có ai để "complain" cho việc code không chạy
- R có thể chiếm dụng hết "available memory".

2. So sánh R với Python

- Sự khác biệt giữa R và Python

Dưới đây là bảng so sánh điểm khác biệt giữa R và Python:

Đối tượng so sánh	R	Python
Mục đích sử dụng	Phân tích và thống kê dữ liệu	Triển khai và sản xuất
Người dùng chính	Học giả và chuyên gia R&D	Lập trình viên và nhà phát triển
Tính linh hoạt	Thư viện có sẵn và dễ sử dụng	Dễ dàng xây dựng các mô hình mới ban đầu
Đường cong học tập	Khó khăn khi bắt đầu	Dễ dàng khi bắt đầu
Tích hợp	Chạy cục bộ	Tích hợp tốt với ứng dụng
Tác vụ	Dễ dàng nhận được kết quả chính	Triển khai thuật toán tốt
Kích thước cơ sở dữ liệu	Xử lý kích thước lớn	Xử lý kích thước lớn
Môi trường phát triển tích hợp (IDE)	Rstudio	Spyder, Ipython Notebook
Ưu điểm	+ Đồ thị được thiết kế đẹp mắt + Danh mục lớn để phân tích dữ liệu	+ Sổ tay Jupyter giúp chia sẻ dữ liệu với đồng nghiệp + Tính toán toán học

	+ Đa dạng, gồm: Giao diện GitHub, RMarkdown, Gói Shiny	+ Triển khai nhanh + Khả năng đọc mã + Tốc độ, vận tốc + Hàm trong Python
Nhược điểm	+ Khó khăn khi mới học + Sự phụ thuộc giữa các thư viện	Không nhiều thư viện như ngôn ngữ lập trình R

- *Nên học R hay Python*

Nếu bạn là người mới bắt đầu tìm hiểu khoa học dữ liệu với nền tảng thống kê, hãy trả lời hai câu hỏi sau:

1. Bạn có muốn tìm hiểu cách hoạt động của thuật toán không?
2. Bạn cũng muốn tìm hiểu về việc triển khai mô hình phải không?

Nếu câu trả lời của bạn cho cả hai câu hỏi là có, bạn nên bắt đầu học **Python** trước. Bởi Python bao gồm các thư viện lý tưởng để thao tác với ma trận hoặc để viết mã các thuật toán. Khi mới bắt đầu, nó có thể dễ dàng hơn khi học cách xây dựng một mô hình từ đầu và sau đó chuyển sang các chức năng từ các thư viện học máy. Nếu bạn muốn làm nhiều việc hơn là thống kê, giả sử như triển khai và khả năng tái tạo, Python là một lựa chọn tốt hơn.

R phù hợp hơn nếu bạn cần viết báo cáo và tạo trang tổng quan. Một điểm nữa là nếu bạn định tập trung vào các phương pháp thống kê đó cũng là một điểm cộng cho R.

Mặt khác, nếu bạn đã biết thuật toán hoặc muốn đi vào phân tích dữ liệu ngay lập tức, thì **cả R và Python** đều phù hợp để bắt đầu. Vậy nên, việc lựa chọn giữa R hoặc Python của bạn phụ thuộc vào:

1. Mục tiêu của sử dụng ngôn ngữ lập trình của bạn là phân tích hay triển khai thống kê?
2. Lượng thời gian bạn có thể đầu tư cho việc học tập.
3. Loại ngôn ngữ nào được sử dụng nhiều nhất ở nơi làm việc của bạn?

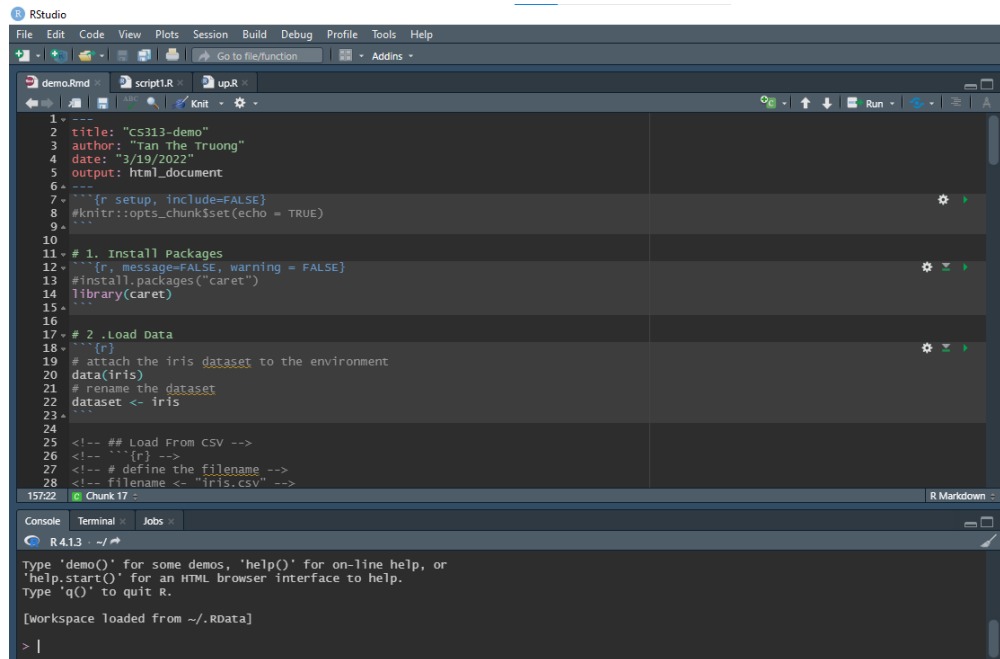
- ***Kết luận***

Tóm lại, khoảng cách giữa R và Python đang ngày càng được thu hẹp. Hầu hết công việc có thể được thực hiện bằng cả hai ngôn ngữ. Hãy lựa chọn ngôn ngữ nào phù hợp với nhu cầu đồng thời cũng là công cụ mà công ty/tổ chức đang sử dụng. Điều này tạo nên sự đồng điệu nơi làm việc và tạo ra năng suất công việc. Sau khi thành thạo ngôn ngữ lập trình đầu tiên, việc học ngôn ngữ thứ hai sẽ trở nên đơn giản hơn.

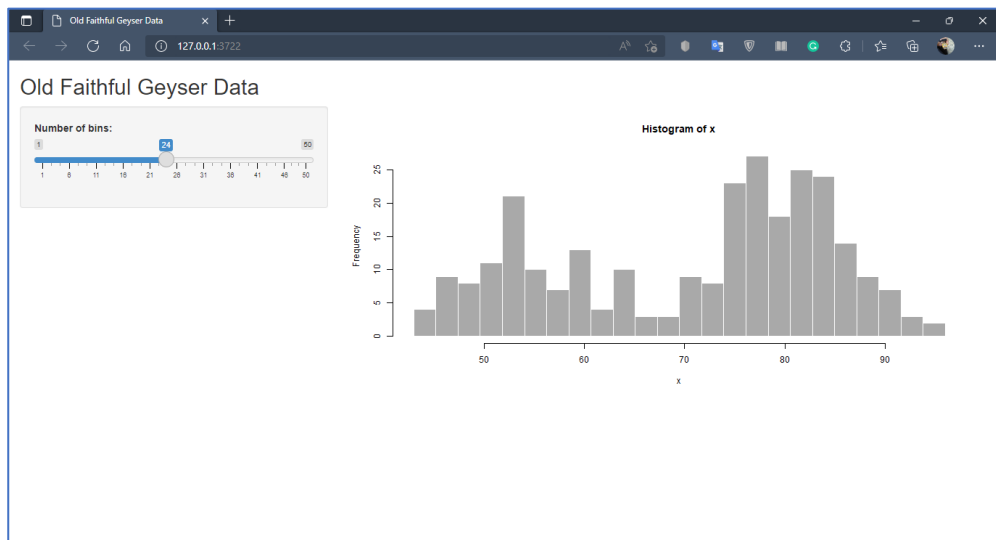
V. Hướng dẫn sử dụng R cơ bản

1. Giao tiếp với R

R có nhiều cách để trình bày và chia sẻ công việc, thông qua tài liệu markdown hoặc một ứng dụng Shiny. Mọi thứ có thể được lưu trữ trên Rpub (khi chắc chắn chia sẻ cộng đồng), GitHub hoặc trang web của doanh nghiệp.



Hình 5. 1. Giao diện của RStudio

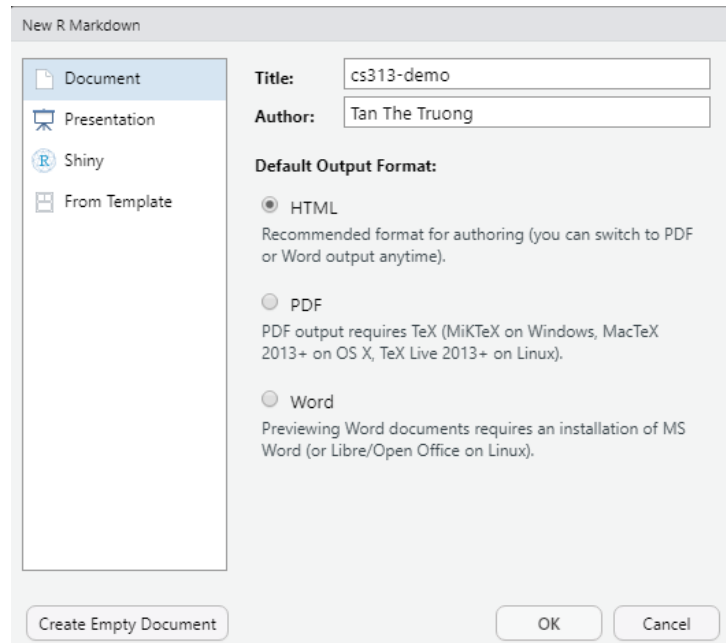


Hình 5. 2. Giao diện web cơ bản của gói shiny

Rstudio chấp nhận markdown để viết tài liệu. Bạn có thể xuất tài liệu ở các định dạng khác nhau:

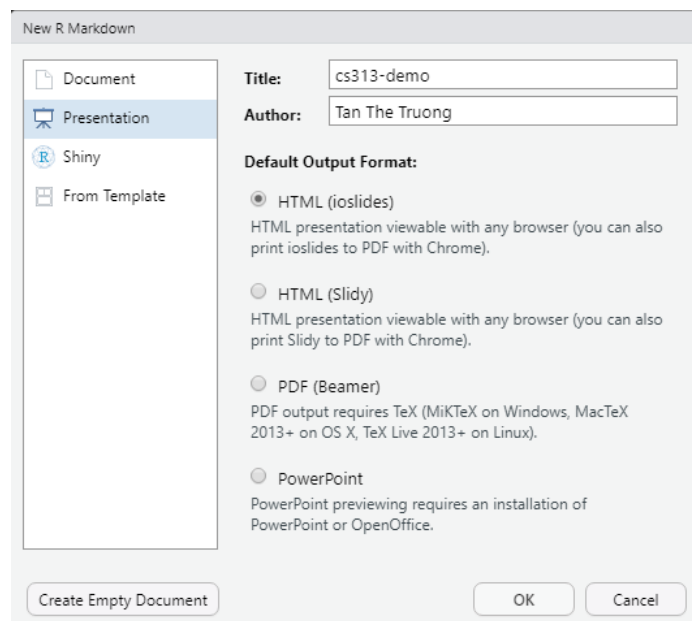
- Tài liệu :

- HTML
- PDF / Latex
- Word



Hình 5. 3. Các tùy chọn tạo Document trong R

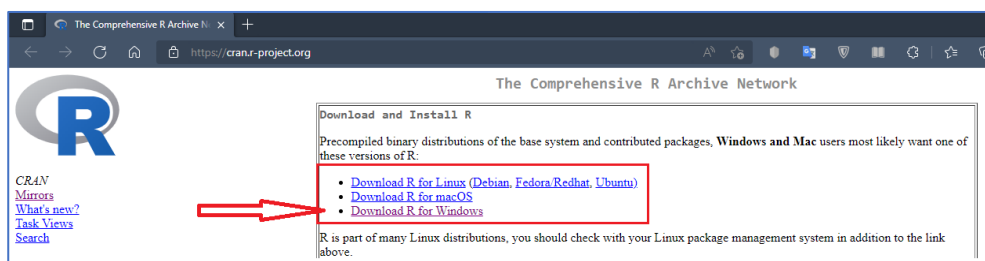
- Bài thuyết trình
 - HTML
 - Trình chiếu PDF
 - PowerPoint



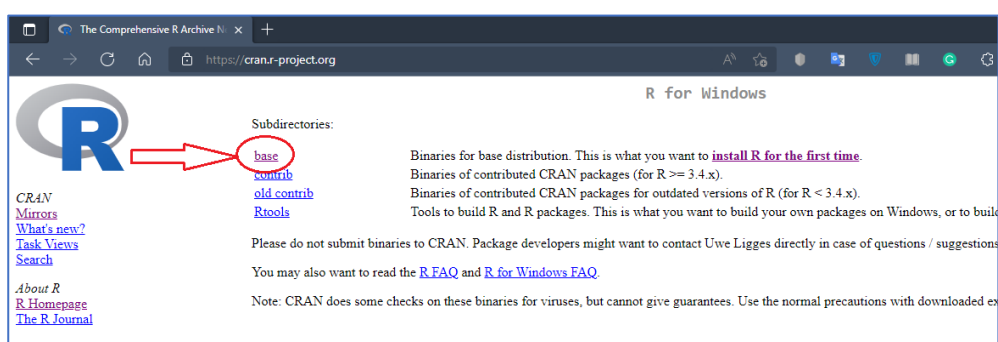
Hình 5. 4. Các tùy chọn tạo bản Presentation trong R

2. Cài đặt R trong window

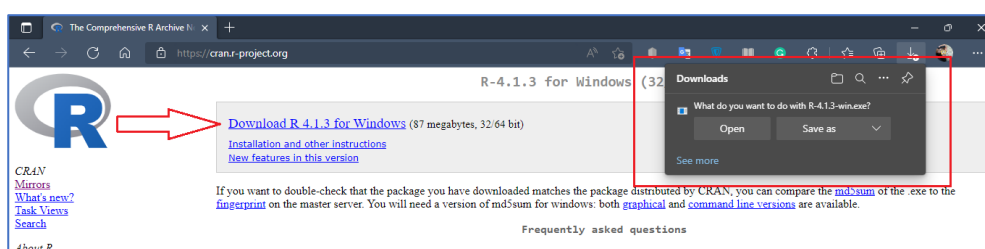
Cài đặt R trên Windows 10 rất đơn giản. Cách dễ nhất là cài đặt nó thông qua CRAN, viết tắt của The Comprehensive R Archive Network (Mạng lưu trữ R toàn diện) [The Comprehensive R Archive Network \(r-project.org\)](https://cran.r-project.org)



Hình 5. 5. Cài đặt R bước 1: chọn hệ điều hành



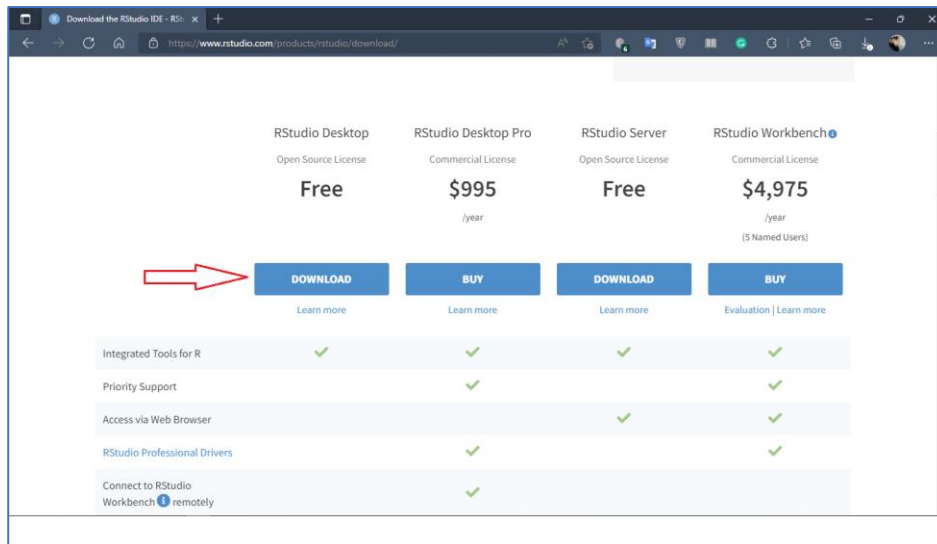
Hình 5. 6. Cài đặt R bước 2: chọn tùy chọn base



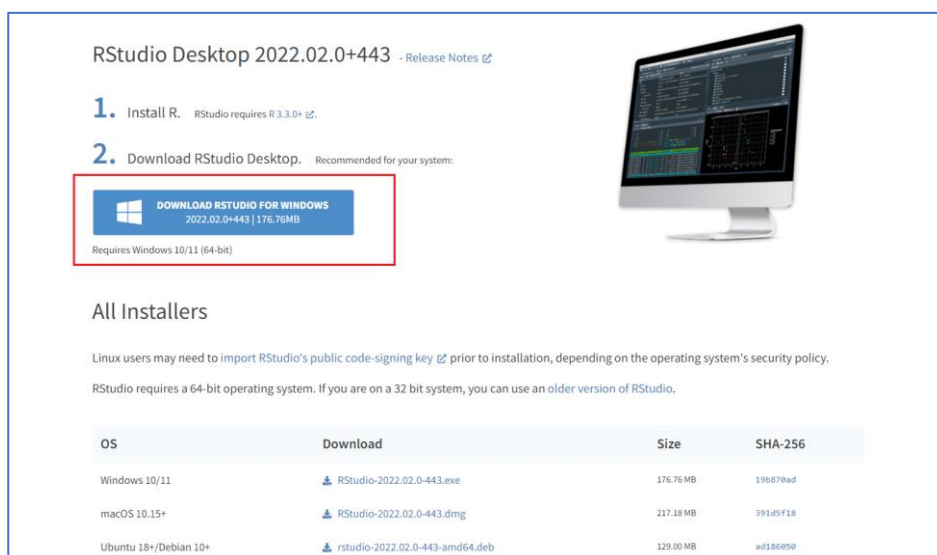
Hình 5. 7. Cài đặt R bước 3: chọn phiên bản mới nhất và tải về

3. Cài đặt RStudio

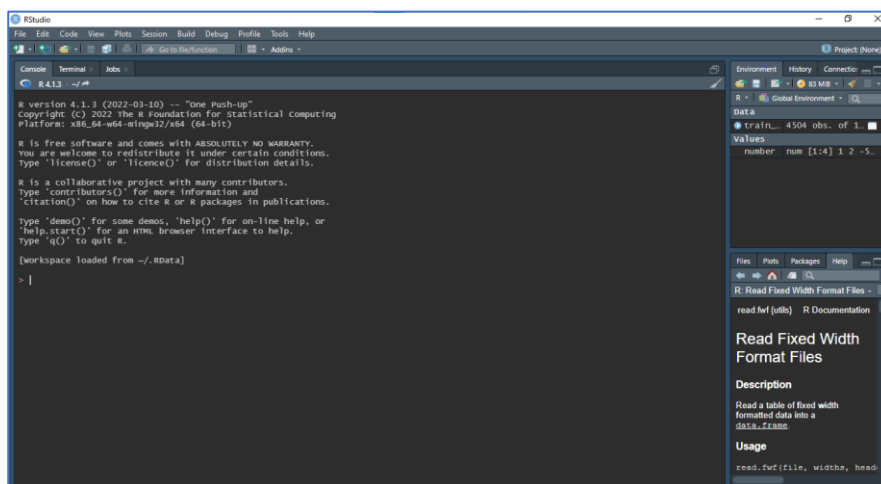
Sau khi R được cài đặt, ta tải RStudio tại [Download the RStudio IDE - RStudio](https://www.rstudio.com/) và tiếp tục cài đặt để có một môi trường được cải thiện nhiều hơn để làm việc trong các tập lệnh R của bạn. Nó bao gồm một bảng điều khiển hỗ trợ thực thi mã trực tiếp và các công cụ để vẽ và theo dõi các biến của bạn trong không gian làm việc, cùng với các tính năng khác.



Hình 5. 8. Cài đặt RStudio bước 1: Các tùy chọn phiên bản của RStudio



Hình 5. 9. Cài đặt RStudio bước 2: lựa chọn phiên bản phù hợp với HĐH và tải về



Hình 5. 10. Cài đặt RStudio: Giao diện với theme tối của RStudio

4. Các thư viện phổ biến trong R

- randomForest (gói randomForest)
- SVM (gói e1071)
- Naviebayes (gói e107)
- KNN (gói mlr3)
- Xgboost (gói mlr3)

4.1. Thư viện randomForest (gói randomForest)

Được dùng cho các bài toán phân loại và hồi quy. RandomForest triển khai thuật toán rừng ngẫu nhiên của Breiman (dựa trên mã Fortran gốc của Breiman và Cutler) để phân loại và hồi quy. Nó cũng có thể được sử dụng ở chế độ không giám sát để đánh giá mức độ gần nhau giữa các điểm dữ liệu.

- Sử dụng:

- Cài đặt packages: `install.packages("randomForest")`
- Tải lên packages: `library(randomForest)`

```
# S3 method for formula
randomForest(formula, data=NULL, ..., subset,
na.action=na.fail)

# S3 method for default
randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500,
             mtry=if (!is.null(y) && !is.factor(y))
             max(floor(ncol(x)/3), 1) else
             floor(sqrt(ncol(x))),
             weights=NULL,
             replace=TRUE, classwt=NULL, cutoff, strata,
             sampsize = if (replace) nrow(x) else
             ceiling(.632*nrow(x)),
             nodesize = if (!is.null(y) && !is.factor(y)) 5
             else 1,
             maxnodes = NULL,
             importance=FALSE, localImp=FALSE, nPerm=1,
             proximity, oob.prox=proximity,
             norm.votes=TRUE, do.trace=FALSE,
```

```

        keep.forest=!is.null(y) && is.null(xtest),
        corr.bias=FALSE,
        keep.inbag=FALSE, ...)

# S3 method for randomForest
print(x, ...)

```

- Tham số

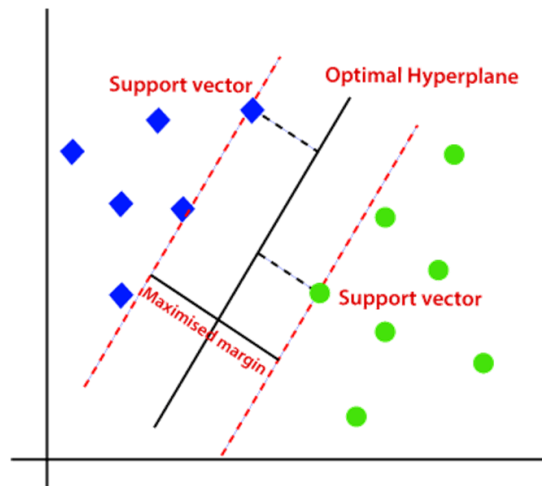
Tham số	Mô tả
data	một khung dữ liệu tùy chọn chứa các biến trong mô hình. Theo mặc định, các biến được lấy từ môi trường mà 'randomForest' được gọi từ đó.
subset	một vector chỉ mục cho biết hàng nào nên được sử dụng. (LƯU Ý: Nếu được đưa ra, đối số này phải được đặt tên.)
na.action	Một hàm để chỉ định hành động sẽ được thực hiện nếu tìm thấy NA. (LƯU Ý: Nếu được đưa ra, đối số này phải được đặt tên.)
x, formula	khung dữ liệu hoặc ma trận các yếu tố dự đoán hoặc công thức mô tả mô hình sẽ được điều chỉnh (cho phương pháp 'print', đối tượng 'randomForest').
y	Một vector phản hồi. Nếu một yếu tố, phân loại được giả định, nếu không, hồi quy được giả định. Nếu bị bỏ qua, 'randomForest' sẽ chạy ở chế độ không được giám sát.
xtest	khung dữ liệu hoặc ma trận (như x) chứa các yếu tố dự đoán cho tập kiểm tra.
ytest	phản hồi cho bộ thử nghiệm
ntree	Số cây cần trồng. Điều này không nên được đặt thành một số quá nhỏ, để đảm bảo rằng mọi hàng đầu vào đều được dự đoán ít nhất một vài lần.

mtry	Số lượng biến được lấy mẫu ngẫu nhiên làm ứng viên tại mỗi lần tách. Lưu ý rằng các giá trị mặc định khác nhau đối với phân loại (sqrt (p) trong đó p là số biến trong 'x') và hồi quy (p / 3)
weights	Một vector có độ dài bằng 'y' là trọng số dương chỉ được sử dụng trong dữ liệu lấy mẫu để phát triển từng cây (không được sử dụng trong bất kỳ phép tính nào khác)
replace	Có nên lấy mẫu các trường hợp có thay thế hay không?
classwt	Trước của các lớp. Không cần cộng lên đến một. Bỏ qua cho hồi quy.
cutoff	(Chỉ phân loại) Một vector có độ dài bằng số lớp. Loại 'winning' cho một quan sát là hạng có tỷ lệ số phiếu bầu trên ngưỡng tối đa. Mặc định là 1 / k trong đó k là số lớp (tức là phần lớn số phiếu thắng cuộc).
strata	Một biến (nhân tố) được sử dụng để lấy mẫu phân tầng.
sampsize	(Các) kích thước của mẫu để vẽ. Đối với phân loại, nếu sampsize là một vector có độ dài là số tầng, thì việc lấy mẫu được phân tầng theo tầng và các phần tử của sampsize cho biết các con số sẽ được rút ra từ các tầng.
nodesize	Kích thước tối thiểu của các nút đầu cuối. Đặt con số này lớn hơn sẽ làm cho cây nhỏ hơn được trồng (và do đó mất ít thời gian hơn). Lưu ý rằng các giá trị mặc định khác nhau đối với phân loại (1) và hồi quy (5).
maxnodes	Số lượng cây nút đầu cuối tối đa trong rừng có thể có. Nếu không được đưa ra, cây sẽ được trồng ở mức tối đa có thể (có giới hạn theo 'nodesize'). Nếu đặt lớn hơn mức tối đa có thể, một cảnh báo sẽ được đưa ra.

importance	Có nên đánh giá tầm quan trọng của các yếu tố dự báo không
localImp	Có nên tính toán thước đo mức độ quan trọng không? (Đặt điều này thành TRUE sẽ ghi đè 'importance'.)
nPerm	Số lần dữ liệu OOB được hoán vị trên mỗi cây để đánh giá mức độ quan trọng của biến. Số lớn hơn 1 cho ước tính ổn định hơn một chút, nhưng không hiệu quả lắm. Hiện chỉ được triển khai cho hồi quy.
proximity	Có nên tính toán độ gần nhau giữa các hàng không?
oob.prox	Có nên chỉ tính toán khoảng cách trên dữ liệu “out-of-bag” không?
norm.votes	Nếu 'TRUE' (mặc định), kết quả cuối cùng của phiếu bầu được biểu thị dưới dạng phân số. Nếu 'FALSE', số phiếu bầu thô được trả về (hữu ích để kết hợp các kết quả từ các lần chạy khác nhau). Bỏ qua cho hồi quy.
do.trace	Nếu được đặt thành 'TRUE', cung cấp đầu ra dài dòng hơn khi chạy 'randomForest'. Nếu được đặt thành một số nguyên, thì đầu ra đang chạy được in cho mọi cây 'do.trace'.
keep.forest	Nếu được đặt thành 'FALSE', rừng sẽ không được giữ lại trong đối tượng đầu ra. Nếu 'xtest' được đưa ra, mặc định là 'FALSE'.
corr.bias	thực hiện hiệu chỉnh sai lệch cho hồi quy?
...	các tham số tùy chọn được chuyển đến hàm mức thấp 'randomForest.default'.

4.2. Thư viện SVM (gói kernLab)

SVM (Support Vector Machine) là một thuật toán học máy có giám sát, được sử dụng chủ yếu để phân loại dữ liệu thành các lớp khác nhau. Không giống như hầu hết các thuật toán, SVM sử dụng một siêu phẳng hoạt động như một ranh giới quyết định giữa các lớp khác nhau.



Hình 5. 11. Mô hình của thuật toán SVM

Nó có thể được sử dụng để thực hiện hồi quy và phân loại chung (kiểu nu và epsilon), cũng như ước tính mật độ. Một giao diện công thức được cung cấp.

- Cách sử dụng:

- Cài đặt packages: `install.packages('e1071')`

```
> # S3 method for formula
> svm(formula, data = NULL, ..., subset, na.action =
na.omit, scale = TRUE)
> # S3 method for default
> svm(x, y = NULL, scale = TRUE, type = NULL, kernel =
"radial", degree = 3, gamma = if (is.vector(x)) 1
else 1 / ncol(x),
coef0 = 0, cost = 1, nu = 0.5,
class.weights = NULL, cachesize = 40, tolerance =
0.001, epsilon = 0.1,
shrinking = TRUE, cross = 0, probability = FALSE,
fitted = TRUE,
..., subset, na.action = na.omit)
```

- **Các đối số:**

Đối số	Mô tả
formula	một mô tả tượng trưng của mô hình để fit.
Data	một khung dữ liệu tùy chọn chứa các biến trong mô hình. Theo mặc định, các biến được lấy từ môi trường mà từ đó 'svm' được gọi.
x	a data matrix, a vector, or a sparse matrix
y	một vector phản hồi có một nhãn cho mỗi hàng/thành phần của x. Có thể là một yếu tố (cho các nhiệm vụ phân loại) hoặc một vector số (cho hồi quy).
scale	Một vector logic cho biết các biến được chia tỷ lệ.
type	<p>'svm' có thể được sử dụng như một máy phân loại, như một máy hồi quy hoặc để phát hiện tính mới. Tùy thuộc vào việc y có phải là một yếu tố hay không, cài đặt mặc định cho 'type' tương ứng là 'C-classification' hoặc 'eps-regression', nhưng có thể bị ghi đè bằng cách đặt một giá trị rõ ràng. Các tùy chọn hợp lệ là:</p> <ul style="list-style-type: none"> • 'C-classification' • 'nu-classification' • 'one-classification' (for novelty detection) • 'eps-regression' • 'nu-regression'
Kernel	<p>hạt nhân được sử dụng trong đào tạo và dự đoán. Bạn có thể cân nhắc thay đổi một số tham số sau, tùy thuộc vào loại hạt nhân.</p> <p>Linear: $(u'v)$</p> <p>polynomial: $((\gamma u'v + \text{coef0})^{\text{degree}})$</p> <p>radial basis: $(e^{(-\gamma u-v ^2)})$</p> <p>sigmoid: $(\tanh(\gamma u'v + \text{coef0}))$</p>

degree	tham số cần thiết cho nhân kiểu 'polynomial' - đa thức (mặc định: 3)
gamma	Tham số cần thiết cho tất cả các hạt nhân ngoại trừ 'linear' (mặc định: $1/(\text{data dimension})$)
Coef0	tham số cần thiết cho nhân kiểu 'polynomial' và 'sigmoid' (mặc định: 0)
cost	
nu	Tham số cần thiết cho 'nu-classification', 'nu-regression', và 'one-classification'
class.weights	một vector trọng số được đặt tên cho các lớp khác nhau, được sử dụng cho các kích thước lớp không đối xứng. Không phải tất cả các mức yếu tố đều phải được cung cấp (trọng lượng mặc định: 1)
cachesize	ộ nhớ đệm tính bằng MB (mặc định 40)
tolerance	dung sai của tiêu chí chấm dứt (mặc định: 0,001)
epsilon	epsilon trong chức năng mất độ nhạy (mặc định: 0,1)
shrinking	tùy chọn có sử dụng heuristics thu nhỏ hay không (mặc định: TRUE)
cross	nếu giá trị số nguyên $k > 0$ được chỉ định, xác thực chéo k lần trên dữ liệu huấn luyện được thực hiện để đánh giá chất lượng của mô hình: tỷ lệ chính xác (accuracy rate) để phân loại và Sai số trung bình bình phương (MSE) cho hồi quy
fitted	logic cho biết liệu các giá trị phù hợp có nên được tính toán và đưa vào mô hình hay không (mặc định: TRUE)
probability	logic cho biết liệu mô hình có cho phép dự đoán xác suất hay không.
...	các tham số bổ sung cho chức năng điều chỉnh mức thấp 'svm.default'

subset	Một vectơ chỉ mục xác định các trường hợp được sử dụng trong mẫu đào tạo
na.action	Một hàm để chỉ định hành động sẽ được thực hiện nếu 'NA' tìm thấy. Hành động mặc định là 'na.omit', dẫn đến việc loại bỏ các trường hợp thiếu giá trị trên bất kỳ biến bắt buộc nào. Một giải pháp thay thế là 'na.fail' gây ra lỗi nếu 'NA' các trường hợp được tìm thấy

4.3. Thư viện Naviebayes (gói mlr3)

- Package 'e107': `install.packages("e107")`
- Thư viện: `library(mlr)`
- `selected_model = makeLearner("classif.naiveBayes")`

```
if (requireNamespace("e1071", quietly = TRUE)) {
  learner = mlr3::lrn("classif.naive_bayes")
  print(learner)
  # available parameters:
  learner$param_set$ids()
}
#> <LearnerClassifNaiveBayes:classif.naive_bayes>
#> * Model: -
#> * Parameters: list()
#> * Packages: mlr3, mlr3learners, e1071
#> * Predict Type: response
#> * Feature types: logical, integer, numeric, factor
#> * Properties: multiclass, twoclass
#> [1] "eps"          "laplace"      "threshold"
```

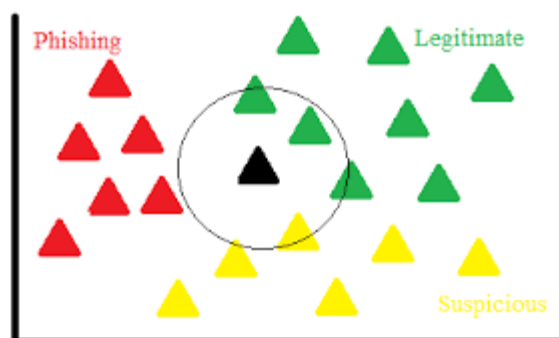
- **Tham số:**

id	Loại	Mặc định	Khoảng	Mô tả

Eps	Numeric	0	$(-\infty, \infty)$	nhân đôi để chỉ định một phạm vi epsilon để áp dụng tính năng làm mịn Laplace (để thay thế xác suất bằng 0 hoặc gần bằng không bởi threshold)
Laplace	Numeric	0	$[0, \infty)$	tích cực điều khiển kép làm mịn Laplace. Giá trị mặc định (0) vô hiệu hóa tính năng làm mịn Laplace.
threshold	Numeric	0.001	$(-\infty, \infty)$	Giá trị thay thế các ô có xác suất trong phạm vi eps.

4.4. Thư viện KNN (gói mlr3)

K - Nearest Neighbors: Là thuật toán phân loại điểm dữ liệu mới dựa trên những điểm dữ liệu gần điểm dữ liệu mới.



Hình 5. 12. Mô hình thuật toán KNN

- Sử dụng:

- Cài đặt các gói

```
+ install.packages("mlr3")
+ install.packages("mlr3learners")
+ install.packages("mlr3tuning")
```

```
+ install.packages("kkn")
```

- Tải lên các gói

```
+ library(mlr3)
```

```
+ library(mlr3learners)
```

```
+ library(mlr3tuning)
```

```
+ library(paradox)
```

- Gọi KNN – Loại mô hình:

```
+ regr.kkn: cho bài toán regression
```

```
+ classif.kkn: cho bài toán classification
```

Ví dụ:

```
learner_diabetes <-
```

```
lrn("classif.kkn", k=10, kernel="rectangular")
```

- Xem các tham số tùy chỉnh của mô hình KNN này

```
as.data.table(lrn("classif.kkn")$param_set)
```

Id	Type	Default	Range	Levels	Mô tả
k	integer	7	[1,∞)	-	Số hàng xóm gần nhất
distance	numeric	2	[0,∞)	-	Tham số khoảng cách Minkowski
kernel	character	optimal	-	rectangular, triangular, epanechnikov, biweight, triweight, cos, inv, gaussian, rank, optimal	Kernel để sử dụng
scale	logical	TRUE	-	TRUE, FALSE	logic, biến tỷ lệ để có sd bằng nhau
ykernel	list	NULL	-	-	Chiều rộng cửa sổ của y-kernel, đặc biệt để dự đoán các lớp thứ tự.

4.5. Thư viện Xgboost (gói mlr3)

- Package ‘xgboost’: `install.packages('xgboost')`

- Gọi gói: `require(xgboost)`

- Tham số: `xgboost(data = NULL, label = NULL, missing = NULL, params = list(), nrounds, verbose = 1, print.every.n = 1L, early.stop.round = NULL, maximize = NULL, ...)`

<code>data</code>	<code>'matrix'</code> , <code>'dgCMatrix'</code> , tệp dữ liệu cục bộ hoặc <code>'xgb.DMatrix'</code> .
<code>label</code>	biến phản hồi. Người dùng không nên đặt trường này, nếu dữ liệu là tệp dữ liệu cục bộ hoặc <code>'xgb.DMatrix'</code> .
<code>missing</code>	Missing chỉ được sử dụng khi đầu vào là ma trận dày đặc, hãy chọn một giá trị thực thể hiện giá trị bị thiếu. Đôi khi dữ liệu sử dụng 0 hoặc giá trị cực trị khác để biểu thị các giá trị bị thiếu.
<code>params</code>	Những cái thường được sử dụng là: <ul style="list-style-type: none"> - Hàm mục tiêu khách quan, những cái chung là + <code>reg:linear</code>: tuyến tính + <code>binary:logistic</code>: logistic để phân loại - <code>eta</code>: kích thước bước của mỗi bước boosting - <code>max.depth</code>: độ sâu tối đa của cây - <code>nthread</code>: số luồng được sử dụng trong đào tạo, nếu không được thiết lập, tất cả các luồng sẽ được sử dụng
<code>nrounds</code>	số lần lặp lại tối đa
<code>verbose</code>	Nếu 0, xgboost sẽ giữ im lặng. Nếu 1, xgboost sẽ in thông tin về hiệu suất. Nếu 2, xgboost sẽ in thông tin cả hiệu suất và thông tin tiến độ thi công
<code>early.stop.round</code>	Nếu NULL, chức năng dừng sớm không được kích hoạt. Nếu được đặt thành số nguyên k, quá trình huấn luyện với tập xác nhận sẽ dừng nếu hiệu suất tiếp tục kém đi trong k vòng.

maximize	Nếu đặt feval và early.stop.round, thì tối đa hóa cũng phải được đặt. tối đa hóa = TRUE có nghĩa là điểm đánh giá càng lớn càng tốt.
...	các tham số khác để chuyển tới 'params'.

5. Cú pháp cơ bản của R

- R phân biệt chữ hoa và chữ thường

Điều này bao gồm tên biến, tên hàm và khá nhiều thứ khác mà bạn có thể nghĩ đến. Ví dụ, tên biến của `myRvariable` không giống với `MyRVariable` và hàm `GetCaseCount()` không thể được viết dưới dạng `getcasecount()`

- Phép gán trong R

Trong R, phép gán được biểu thị theo ba cách:

Phép gán	Ý nghĩa	Ví dụ
=	Gán đơn giản	<code>Var1 = "one"</code>
<-	Gán sang trái	<code>Var2 <- "two"</code>
->	Gán sang phải	<code>Var3 -> "three"</code>

Lưu ý:

1. Nếu chúng ta có dấu cách giữa dấu < và - thì R không thực hiện phép gán, và báo phép gán không hiệu lực bằng giá trị FALSE. Ví dụ:

```
> x <- 4
> x < - 5 # phép gán không hiệu lực vì có dấu cách giữa < và -
[1] FALSE # báo là FALSE
> x
[1] 4      # x vẫn bằng 4, không phải là
```

2. Các lệnh R được kết thúc bằng dấu chấm phẩy (;) hoặc dòng mới, dòng tiếp tục không yêu cầu ký tự đặc biệt hoặc thụt lề, ví dụ như sau:

```
var1 <- var2 +
3
```

Câu lệnh trên được đọc là `var1 <- var2 + 3`, vì R tiếp tục đọc đầu vào cho đến khi một lệnh hoàn tất về mặt cú pháp. Tuy nhiên:

```
var1 <- var2
```

```
+3
```

Câu lệnh sẽ được đọc dưới dạng hai lệnh riêng biệt và `var1` sẽ được đặt thành giá trị của `var2`.

- ***Nhóm các câu lệnh được chỉ định bằng dấu ngoặc nhọn***

Các nhóm câu lệnh trong cấu trúc như vòng lặp, biểu thức điều kiện và hàm được chỉ ra bằng cách đặt các câu lệnh trong dấu ngoặc nhọn, ví dụ:

```
> while (!spssdata.IsLastSplit()) {  
    data <- spssdata.GetSplitDataFromSPSS()  
    cat("\nCases in Split: ", length(data[,1]))  
}
```

- ***c là để kết hợp (hoặc nối, chuyển đổi/ ép kiểu)***

Khi bạn tạo một mảng trong hầu hết các ngôn ngữ lập trình, cú pháp sẽ giống như sau:

```
myArray = array (1, 1, 2, 3, 5, 8);  
Hoặc: int myArray = {1, 1, 2, 3, 5, 8};  
Hoặc có thể: myArray = [1, 1, 2, 3, 5, 8]
```

Tuy nhiên, trong R, có một phần bổ sung: Để đặt nhiều giá trị vào một biến duy nhất, bạn sử dụng hàm `c()`, chẳng hạn như:

```
my_vector <- c(1, 1, 2, 3, 5, 8)
```

Đây là một lỗi sai gặp khá nhiều khi bạn mới tiếp cận với ngôn ngữ này.

Trong đa số trường hợp, `c` là bắt buộc. Tuy nhiên cũng có ngoại lệ như khi bạn muốn một dãy số liên tiếp nhau từ 1 đến 10:

```
my_vector <- (1:10)
```

lúc này có hay không có `c` đều là đúng.

Một điểm quan trọng nữa về hàm `c()`: Nó giả định rằng mọi thứ trong vector của bạn có cùng kiểu dữ liệu - nghĩa là tất cả các số hoặc tất cả các ký tự. Nếu bạn tạo một vector chẳng hạn như:

```
my_vector <- c(1, 4, "hello", TRUE)  
[1] "1" "4" "hello" "TRUE"
```

Ta sẽ không có một vector có hai đối tượng số nguyên, một đối tượng ký tự và một đối tượng logic. Thay vào đó, `c()` sẽ làm những gì có thể để chuyển tất cả chúng thành tất cả cùng một kiểu đối tượng, các giá trị không phải ký tự bị ép buộc thành kiểu ký tự nếu một trong các phần tử là ký tự, trong trường hợp này tất cả được ép thành các đối tượng ký tự. Ta cũng có thể nghĩ về `c()` như cho "chuyển đổi" hoặc "ép kiểu".

Để tạo một bộ sưu tập với nhiều kiểu đối tượng, bạn cần một danh sách R, không phải một vector. Bạn tạo một danh sách bằng hàm `list()`, không phải `c()`, chẳng hạn như:

```
My_list <- list(1, 4, "hello", TRUE)
```

- ***Chỉ mục vector trong R bắt đầu từ 1, không phải 0***

Trong hầu hết các ngôn ngữ máy tính, mục đầu tiên trong vector, danh sách hoặc mảng là mục 0. Trong R, đó là mục 1. `my_vector[1]` là mục đầu tiên trong `my_vector`. Một khi đã quen với nó, có thể sẽ nhận ra nó vô cùng tiện lợi và trực quan, và tự hỏi tại sao nhiều ngôn ngữ hơn không sử dụng hệ thống thân thiện với con người hơn này. Rốt cuộc, mọi người đếm mọi thứ bắt đầu từ 1, không phải 0.

- ***R quy ước trích dẫn***

Các chuỗi trong ngôn ngữ lập trình R có thể được đặt trong dấu nháy đơn (`'`) hoặc dấu nháy kép (`"`).

Để chỉ định một dấu nháy đơn (`'`) trong một chuỗi, hãy đặt chuỗi trong dấu ngoặc kép. Ví dụ, `"Joe's Bar and Grille"` được coi là Joe's Bar và Grille.

Để chỉ định dấu ngoặc kép (`"`) trong một chuỗi, hãy sử dụng dấu ngoặc đơn để bao quanh chuỗi lớn. Ví dụ: `'Categories Labeled "UNSTANDARD" in the Report'`

Trong ngôn ngữ lập trình R, dấu nháy kép cùng loại với dấu nháy bên ngoài không được phép. Ví dụ, `'Joe''s Bar and Grille'` dẫn đến một lỗi.

Nhận xét đơn được viết bằng `#` ở đầu câu lệnh như sau

```
# My first program in R Programming
```

R không hỗ trợ nhận xét nhiều dòng nhưng bạn có thể thực hiện một thủ thuật như sau:

```
if(FALSE) {  
    "This is a demo for multi-line comments and it  
    should be put inside either a  
    single OR double quote"  
}
```

- **Đường dẫn các tệp**

Vì các “escape sequence” (là sự kết hợp của các ký tự có ý nghĩa đặc biệt) trong ngôn ngữ lập trình R bắt đầu bằng dấu gạch chéo ngược (\) - chẳng hạn như “\n” cho dòng mới và “\t” cho tab - nên ta sử dụng dấu gạch chéo lên (/) trong đường dẫn tệp trên Windows.

```
spssRGraphics.Submit("/temp/R_graphic.jpg")
```

Ngoài ra, ta có thể thay từng dấu gạch chéo ngược bằng 1 cặp dấu gạch chéo ngược khác, như ví dụ sau:

```
spssRGraphics.Submit("\\temp\\R_graphic.jpg")
```

- **Từ khóa trong R**

Các từ dành riêng trong lập trình R là một tập hợp các từ có ý nghĩa đặc biệt và không thể được sử dụng như một định danh (tên biến, tên hàm, v.v.).

Đây là danh sách các từ dành riêng trong trình phân tích cú pháp R.

Từ dành riêng cho R				
if	else	repeat	while	function
for	in	next	break	TRUE
FALSE	NULL	Inf	NaN	NA
NA_integer_	NA_real_	NA_complex_	NA_character_	...

Danh sách trên có thể được xem bằng cách gõ `help(reserved)` hoặc `?reserved` tại dấu nhắc lệnh R như sau.

```
> ?reserved
```


- Trong số các từ này,,,,, `if` và `else` được sử dụng cho các điều kiện `repeat`, vòng lặp và các hàm do người dùng xác định. `while` function for in next break. Chúng tạo thành các khối xây dựng cơ bản của lập trình trong R.
- `TRUE` và `FALSE` là các hằng logic trong R.
- `NULL` đại diện cho sự vắng mặt của một giá trị hoặc một giá trị không xác định.
- `Inf` là cho "Infinity" (vô cực), ví dụ khi 1 chia cho 0 trong khi `NaN` "Not a Number" (Không phải số), ví dụ: khi 0 chia cho 0.
- `NA` là viết tắt của "Not Available" (Không có sẵn) và được sử dụng để biểu thị các giá trị bị thiếu.
- R là một ngôn ngữ phân biệt chữ hoa chữ thường. Có nghĩa là `TRUE` và `True` không giống nhau.

```
> TRUE <- 1
Error in TRUE <- 1 : invalid (do_set) left-hand
side to assignment
> True <- 1
> TRUE
[1] TRUE
> True
[1] 1
```

6. Tutorial R

6.1. Toán tử R

Các toán tử được sử dụng để thực hiện các hoạt động trên các giá trị và biến. Các toán tử R được phân thành sáu loại khác nhau:

- *Toán tử số học:* `+` , `-` , `*` , `/` , `%%` (chia lấy dư) , `^` (mũ) , `%/%` (chia lấy nguyên)
- *Toán tử so sánh:* `==` , `!=` , `>` , `<` , `>=` , `<=`
- *Toán tử logic:* `&&` , `||` , `!`
- *Toán tử logic thao tác từng phần tử:* Các toán tử này được sử dụng để thực hiện các phép toán logic trên vector theo cách thức từng phần tử.
`+` & `:` Trả về `True` nếu các phần tử tương ứng của cả hai vector đều đúng

+ | : Trả về True nếu một trong các phần tử tương ứng của cả hai vector là true

```
> v1 <- c(0,1,2,3,4)
> v2 <- c(0,1,0,1,0)
> # and
> v1 & v2
[1] FALSE TRUE FALSE TRUE FALSE
> # or
> v1 | v2
[1] FALSE TRUE TRUE TRUE TRUE
```

- *Toán tử thành viên*: Toán tử thành viên được sử dụng để kiểm tra xem một mục cụ thể có trong vector hoặc danh sách hay không.

+ %in% : Trả về True nếu một giá trị có trong vector hoặc danh sách

```
> v <- list("red", "green", "blue")
> "red" %in% v
[1] TRUE
> "blue" %in% v
[1] TRUE
```

- *Toán tử gán*: =, <- (gán trái), -> (gán phải)

6.2. Cấu trúc dữ liệu R

- **R Vector**

vector là một những kiểu dữ liệu cơ bản của R. Vector là một tập hợp các phần tử, tất cả đều cùng kiểu (tương tự như một mảng trong các ngôn ngữ lập trình khác nhưng linh hoạt hơn). Để hiểu rõ vector của R các bạn có thể đọc tài liệu chi tiết về vector bằng lệnh:

```
?vector
```

+ *Tạo vector*: Trong R, có nhiều cách để tạo một vector mới, cách đơn giản nhất là dùng hàm `c()`.

```
> # integer vector
> c(1, 2, 3, 4, 5, 6)
[1] 1 2 3 4 5 6
> # double vector
> c(1*pi, 2*pi, 3*pi, 4*pi)
```

```
[1] 3.141593 6.283185 9.424778 12.566371
> # character vector
> c("red", "green", "blue")
[1] "red" "green" "blue"
> # logical vector
> c(TRUE, FALSE, TRUE, FALSE)
[1] TRUE FALSE TRUE FALSE
```

Một số cách khác tạo vector

- Sử dụng toán tử :

Ta có thể tạo một dãy số cách đều nhau bằng toán tử ' : '

```
> # sequence of numbers from 1 to 10
> 1:10
[1] 1 2 3 4 5 6 7 8 9 10
```

- Sử dụng hàm **seq()**

Hàm `seq()` hoạt động giống như toán tử ' : ', ngoại trừ bạn có thể chỉ định một số gia khác (kích thước bước).

```
> # sequence of numbers from 1 to 10 with
  increment of 2
> seq(from=1,to=10,by=2)
[1] 1 3 5 7 9
```

- Sử dụng hàm **rep()**

Với hàm `rep()`, bạn có thể tạo một chuỗi bằng cách lặp lại các giá trị nhất định.

```
> # repeat a vector 3 times
> rep(x=c(1,2,3),times=3)
[1] 1 2 3 1 2 3 1 2 3
```

+ Một số hàm xử lý trong vector:

Mô tả	Ví dụ
Chuyển kiểu dữ liệu của vector	<code>v <- c(0, 1, 2, 3, 4, 5)</code> <code>as.vector(v, mode="character")</code>

Đặt tên cho phần tử trong vector. Điều này cho phép truy vấn phần tử trong vector bằng tên	<pre>v <- c("Apple", "Banana", "Cherry") names(v) <- c("A", "B", "C")</pre>
Trích giá trị từ vector	<pre>v <- c("a", "b", "c", "d", "e", "f") v[1]</pre>
Thay đổi giá trị của phần tử trong vector	<pre>v <- c("a", "b", "c", "d", "e", "f") v[3] <- 1</pre>
Thêm phần tử vào vector	<pre>v <- c(1, 2, 3) v <- c(v, 4)</pre>
Sắp xếp vector	<pre>sort(v, decreasing=FALSE)</pre>
Lấy tổng số phần tử trong vector	<pre>length(v)</pre>

▪ R Matrix

Ma trận là một tập hợp các phần tử, tất cả đều cùng kiểu, được sắp xếp theo bố cục hai chiều. Tóm lại, ma trận chỉ là một vector có hai chiều.

+ *Tạo ma trận*: Tạo ma trận bằng cách sử dụng hàm `matrix()` và chỉ định dữ liệu cũng như số hàng và cột để tạo ma trận.

```
> # Create a numeric matrix
> m <- matrix(1:6, nrow=2, ncol=3)
> m
      [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Theo mặc định, ma trận được điền theo từng cột. Bằng cách đặt `byrow=TRUE`, bạn có thể điền vào từng hàng (*row-by-row*) của ma trận.

```
> # Create a numeric matrix row-by-row
> m <- matrix(1:6, nrow=2, ncol=3, byrow=TRUE)
> m
      [,1] [,2] [,3]
[1,]    1    3    5
```

[2,]	2	4	6
-------	---	---	---

+ Một số hàm xử lý trong ma trận

Chức năng	Ví dụ
Liên kết các vector thành ma trận theo hàng	<code>rbind(v1, v2, v3)</code>
Liên kết các vector thành ma trận theo cột	<code>cbind(v1, v2, v3)</code>
Lấy kích thước ma trận, thay đổi kích thước ma trận m	<code>dim(m), dim(m) <- c(3,2)</code>
Đặt tên cho hàng, cột của ma trận	<code>dimnames(m) <- list(c("r1", "r2"), c("c1", "c2", "c3"))</code>
Đặt tên cho hàng của ma trận	<code>rownames(m) <- c("r1", "r2")</code>
Đặt tên cho cột của ma trận	<code>colnames(m) <- c("c1", "c2", "c3")</code>
Đọc giá trị phần tử trong mảng	<code>m[2, 3]</code>
Lấy tổng số phần tử trong ma trận	<code>length(m)</code>

▪ R List

Vector và ma trận là cấu trúc dữ liệu cực kỳ hữu ích trong R, nhưng chúng có một hạn chế khác biệt: chúng chỉ có thể lưu trữ một loại dữ liệu.

Tuy nhiên, List có thể lưu trữ nhiều loại giá trị cùng một lúc. Một danh sách có thể chứa một ma trận số, một vector logic, một chuỗi ký tự, một đối tượng nhân tố và thậm chí một danh sách khác.

+ *Tạo list*: Tạo một danh sách cũng giống như tạo một vector; chỉ cần chuyển một chuỗi các phần tử được phân tách bằng dấu phẩy vào hàm **list()**.

```
> # A list of mixed datatypes
> lst <- list(1, "abc", 1.23, TRUE)
```

Cách tốt nhất để hiểu nội dung của danh sách là sử dụng hàm cấu trúc `str()`. Nó cung cấp một màn hình nhỏ gọn về cấu trúc bên trong của một danh sách.

```
> lst <- list(1, "abc", 1.23, TRUE)
> str(lst)
List of 4
 $ : num 1
 $ : chr "abc"
 $ : num 1.23
 $ : logi TRUE
```

Ghi chú: Hàm `str()` cung cấp một màn hình nhỏ gọn về cấu trúc bên trong của bất kỳ đối tượng R nào.

+ *Một số xử lý trong list*

Chức năng	Ví dụ
Lập danh sách con theo vị trí + <code>[]</code> : trả về một danh sách với các phần tử đã chọn + <code>[[]]</code> : trả về chính xác phần tử đã chọn	<pre>> lst <- list(1, "abc", > 1.23, TRUE, 1:3) # extract 2nd element > lst[2] [[1]] [1] "abc" > lst[[2]] [1] "abc"</pre>
Lập danh sách con theo tên + Ta có thể truy cập từng phần tử bằng cách chỉ định tên của nó trong <code>[[]]</code> hoặc sử dụng toán tử <code>\$</code> .	<pre>> months <- list(JAN=1, FEB=2, MAR=3, APR=4) > # extract element by its name > months[["MAR"]] [1] 3 > # same as above but using the \$ operator > months\$MAR</pre>

	<pre>[1] 3</pre>
<p>Sửa đổi phần tử trong danh sách</p> <p>+ sử dụng <code>[[]]</code> hoặc <code>\$</code> để truy cập phần tử đó và chỉ cần gán một giá trị mới.</p>	<pre>> # Modify 3rd list element > lst <- list("a", "b", "c", "d", "e", "f") > lst[[3]] <- 1 > lst[[3]] [1] 1</pre>
<p>Thêm phần tử vào một danh sách</p> <p>+ Nếu phần tử đã có trong danh sách, nó sẽ được cập nhật khác, một phần tử mới sẽ được thêm vào danh sách.</p>	<pre>> lst <- list(1, 2, 3) > lst[[4]] <- 4 > lst [1] 1 2 3 4 > lst <- append(lst, c("a", "b", "c")) > lst [1] 1 2 3 4 "a" "b" "c"</pre>
<p>Xóa một phần tử khỏi danh sách</p> <p>+ chọn nó theo vị trí hoặc theo tên, rồi gán NULL cho nó.</p> <p>+ Sử dụng <code>[]</code>, bạn có thể xóa nhiều thành phần cùng một lúc.</p>	<pre>> lst <- list("a", "b", "c", "d", "e") > lst[[3]] <- NULL > lst [1] a" "b" "d" "e" > lst[1:3] <- NULL > lst [1] "e"</pre>
<p>Kết hợp danh sách</p> <p>+ <code>c()</code> không chỉ tạo ra các vector. Nó cũng có thể được sử dụng để kết hợp các danh sách thành một danh sách mới.</p>	<pre>> lst1 <- list("a", "b", "c") > lst2 <- list(1, 2, 3) > lst <- c(lst1, lst2) > lst [1] "a" "b" "c" 1 2 3</pre>
<p>Chuyển danh sách thành vector</p>	<pre># Flatten the list into a vector and compute mean</pre>

+ sử dụng hàm <code>unlist()</code>	<pre>> lst <- list(5, 10, 15, 20, 25) > class(unlist(lst)) [1] "vector"</pre>
Độ dài của danh sách + sử dụng hàm <code>length()</code>	<pre>> lst <- list(5, 10, 15, 20) > length(lst) [1] 4</pre>

▪ R Data Frame

Data Frame - Khung dữ liệu là danh sách các vector có độ dài bằng nhau. Cách dễ nhất để nghĩ về một khung dữ liệu là như một trang tính Excel.

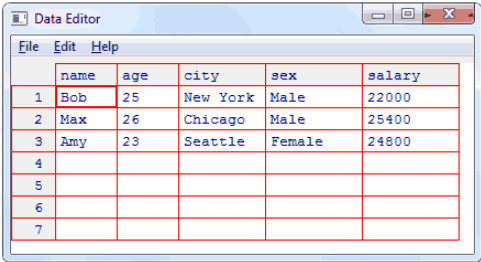
+ *Tạo data frame*: Sử dụng hàm **`data.frame()`**

```
> # Create a data frame to store employee records
> name <- c("Bob", "Max", "Sam")
> age <- c(25, 26, 23)
> city <- c("New York", "Chicago", "Seattle")
> df <- data.frame(name, age, city)
> df
  name age  city
1  Bob  25 New York
2  Max  26  Chicago
3  Sam  23  Seattle
```

+ *Một số xử lý trong data frame*

Chức năng	Ví dụ
<p>Hàm <code>subset()</code>:</p> <pre>subset(df, select, subset)</pre> <p>df: Khung dữ liệu bạn muốn tập hợp con.</p> <p>select: Một tên cột hoặc một vector tên cột sẽ được chọn.</p>	<pre>> # select the employee name and city with age > 22 > subset(df, select=c(name, city), subset=(age > 22)) name city 2 Max Houston 5 Sam Chicago</pre>

subset: Một biểu thức logic chọn các hàng.	
<p>Thêm hàng và cột vào data frame</p> <p>+ Thêm cột: hàm <code>cbind()</code></p> <p>+ Thêm hàng: hàm <code>rbind()</code></p>	<pre>> row <- data.frame(name = "Sam", age = 22, city = "New York") > rbind(df, row) > sex <- factor(c("M", "M", "F", "G")) > cbind(df, sex) name age city sex 1 Bob 25 New York M 2 Max 26 Chicago M 3 Amy 23 Seattle F 4 Sam 22 New York G</pre>
<p>Kết hợp hai data frame</p> <p>+ kết hợp theo cột: <code>cbind()</code></p> <p>+ kết hợp theo hàng: <code>rbind()</code></p> <pre>> df1 name age city 1 Bob 25 New York > df2 name age city 1 Eve 21 Chicago > df3 sex salary 1 Male 22000</pre>	<pre>> cbind(df1, df3) name age city sex salary 1 Bob 25 New York Male 22000 > rbind(df1, df2) name age city 1 Bob 25 New York 2 Eve 21 Chicago</pre>
<p>Hợp nhất data frame theo cột chung</p> <p>+ sử dụng hàm <code>merge()</code></p> <p># Merge two data frames by common column 'name'</p> <pre>> df1 name age city 1 Bob 25 New York</pre>	<pre>> merge(df1, df2, by="name") name age city sex salary 1 Amy 23 Seattle Female 24800 2 Bob 25 New York Male 22000</pre>

<pre>2 Amy 23 Seattle > df2 name sex salary 1 Amy Female 24800 2 Bob Male 22000</pre>	
<p>Sửa đổi data frame</p> <p>+ Truy cập phần tử bằng toán tử <code>[]</code> và chỉ cần gán một giá trị mới.</p>	<pre>> df name age city 1 Bob 25 New York 2 Max 26 Chicago > # modify 2nd column > df[2] <- c(23,21) > df name age city 1 Bob 23 New York 2 Max 21 Chicago</pre>
<p>Chỉnh sửa data frame</p> <p>+ <code>edit()</code> : Nó mở ra trình chỉnh sửa dữ liệu hiển thị khung dữ liệu của bạn trong một cửa sổ giống như bảng tính. Ghi đè chỉnh sửa dùng lệnh <code>df <- temp</code></p> <p>+ <code>fix()</code> : hoạt động giống như <code>edit()</code> ngoại trừ nó ghi đè khung dữ liệu sau khi bạn đóng trình chỉnh sửa.</p>	<pre>> df name age city sex salary 1 Bob 25 New York Male 22000 2 Max 26 Chicago Male 25400 3 Amy 23 Seattle Female 24800 > temp <- edit(df) > df <- temp</pre>  <pre>> fix(df)</pre>
<p>Sắp xếp một data frame</p> <p>+ hàm <code>order()</code></p> <pre>> # Sort the data frame by age > df</pre> <pre> name age city 1 Bob 25 New York</pre>	<pre>> # sort in ascending order > df[order(df\$age),] name age city 3 Amy 23 Seattle 1 Bob 25 New York 2 Max 26 Chicago</pre>

2	Max	26	Chicago
3	Amy	23	Seattle

▪ R Factor

Trong các vấn đề thực tế, ta thường gặp phải dữ liệu có thể được phân loại trong các danh mục (ví dụ giới tính, màu sắc,...). Để lưu trữ dữ liệu phân loại như vậy, R có một cấu trúc dữ liệu đặc biệt được gọi là **factor**. Một factor là một bộ sưu tập các mặt hàng có thứ tự. Các giá trị khác nhau mà yếu tố có thể nhận được gọi là **levels**.

+ Tạo factor: sử dụng hàm `factor()`

```
> # Factor storing hair color values
> hcolors <- c("Blonde", "Black", "Brown", "Red")
> f <- factor(hcolors)
> f
[1] Blonde Black  Brown  Red
Levels: Black Blonde Brown Red
```

+ Một số xử lý trong factor

Chức năng	Ví dụ
Hiển thị tất cả cấp độ (levels) của factor + sử dụng hàm <code>levels()</code>	<pre>> gender <- c("Female", "Male", "Female") > f <- factor(gender) > levels(f) [1] "Female" "Male"</pre>
Nhãn của factor + R cho phép gán tên cho các cấp độ (levels) bằng cách thêm đối số <code>label</code> trong hàm <code>factor()</code>	<pre>> gender <- c("Female", "Male", "Female") > f <- factor(gender, levels=c("Male", "Female"), labels=c("M", "Fe")) > f [1] Fe M Fe Levels: M Fe</pre>
Các yếu tố có thứ tự trong factor	<pre># argument ordered=TRUE in factor() # Create ordinal levels</pre>

+ Loại dữ liệu có thể sắp xếp theo thứ tự hoặc quy mô được gọi là dữ liệu thứ tự (Ordinal data). Trong R, có một kiểu dữ liệu đặc biệt cho dữ liệu thứ tự. Loại này được gọi là các yếu tố có thứ tự.	<pre>> record <- c("win", "tie", "loss", "tie", "loss") > f <- factor(record, ordered = TRUE) > f [1] win tie loss tie loss Levels: loss < tie < win</pre>
Xóa mức độ trong factor khi không sử dụng + sử dụng hàm <code>droplevels()</code>	<pre>> droplevels(f) [1] win loss loss Levels: loss win</pre>
Tóm tắt một factor + sử dụng hàm <code>summary()</code>	<pre>> summary(f) win loss 1 2</pre>

6.3. Cấu trúc điều khiển

Nguyên lý hoạt động tương tự như các ngôn ngữ như C++ hay python.

Cấu trúc điều khiển	Ví dụ
if...else	<pre>> x <- 0 > if (x < 0) { print("Negative number") } > else if (x > 0) { print("Positive number") } > else print("Zero") [1] "Zero"</pre>
switch (expression, list)	<pre>> switch(2, "red", "green", "blue") [1] "green" > x <- switch(4, "red", "green", "blue") > x NULL</pre>
while()	<pre>> i <- 1 > while (i < 4) { print(i) i = i+1 } [1] 1 [1] 2</pre>

	<code>[1] 3</code>
<code>for()</code>	<pre>> x <- c(2,5,3,9,8,11,6) > count <- 0 > for (val in x) { if(val %% 2 == 0) count = count+1 } > print(count) [1] 3</pre>
<code>repeat()</code>	<pre>> x <- 1 > repeat { print(x) x = x+1 if (x == 3){ break } } [1] 1 [1] 2</pre>

6.4. Hàm

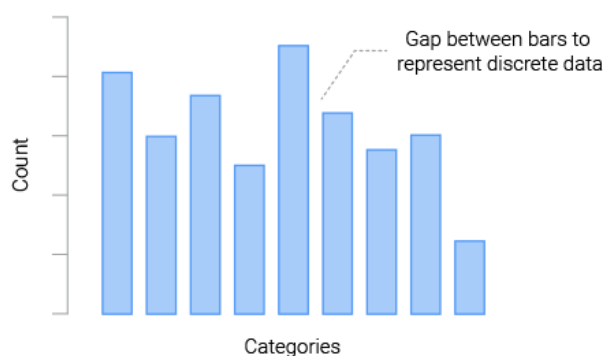
Về cơ bản, hàm trong R có các vấn đề, chức năng tương tự như các ngôn ngữ lập trình là C++ hoặc python.

```
> sum <- function(x, y=3) {
  return(x + y)
}
> sum(2, 3)
> sum(4)

# output
[1] 5
[2] 7
```

6.5. Đồ họa cơ sở R

- R Bar plot – biểu đồ thanh ngang

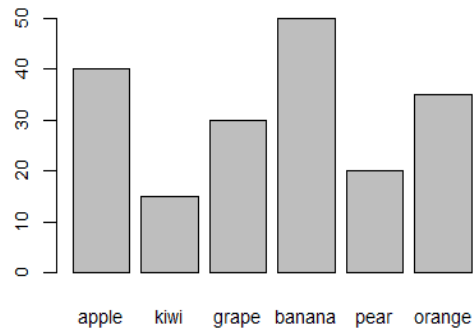


Hình 5. 13. Mô hình biểu đồ thanh ngang - Bar plot trong R

Biểu đồ thanh (Bar Graph hoặc Bar Chart) là một màn hình hiển thị dữ liệu bằng đồ họa sử dụng các thanh có độ cao khác nhau.

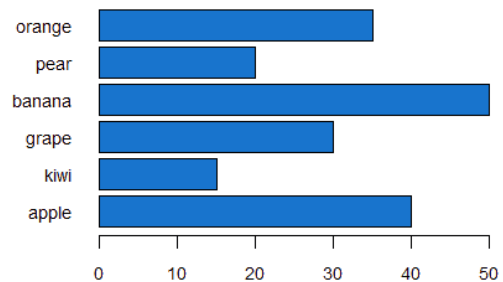
```
> survey <- c(apple=40, kiwi=15, grape=30,
banana=50, pear=20, orange=35)
> survey
apple kiwi grape banana pear orange
40    15    30    50    20    35
```

```
> barplot(survey)
```



+ Horizontal Bar Graph – Đồ thị thanh ngang

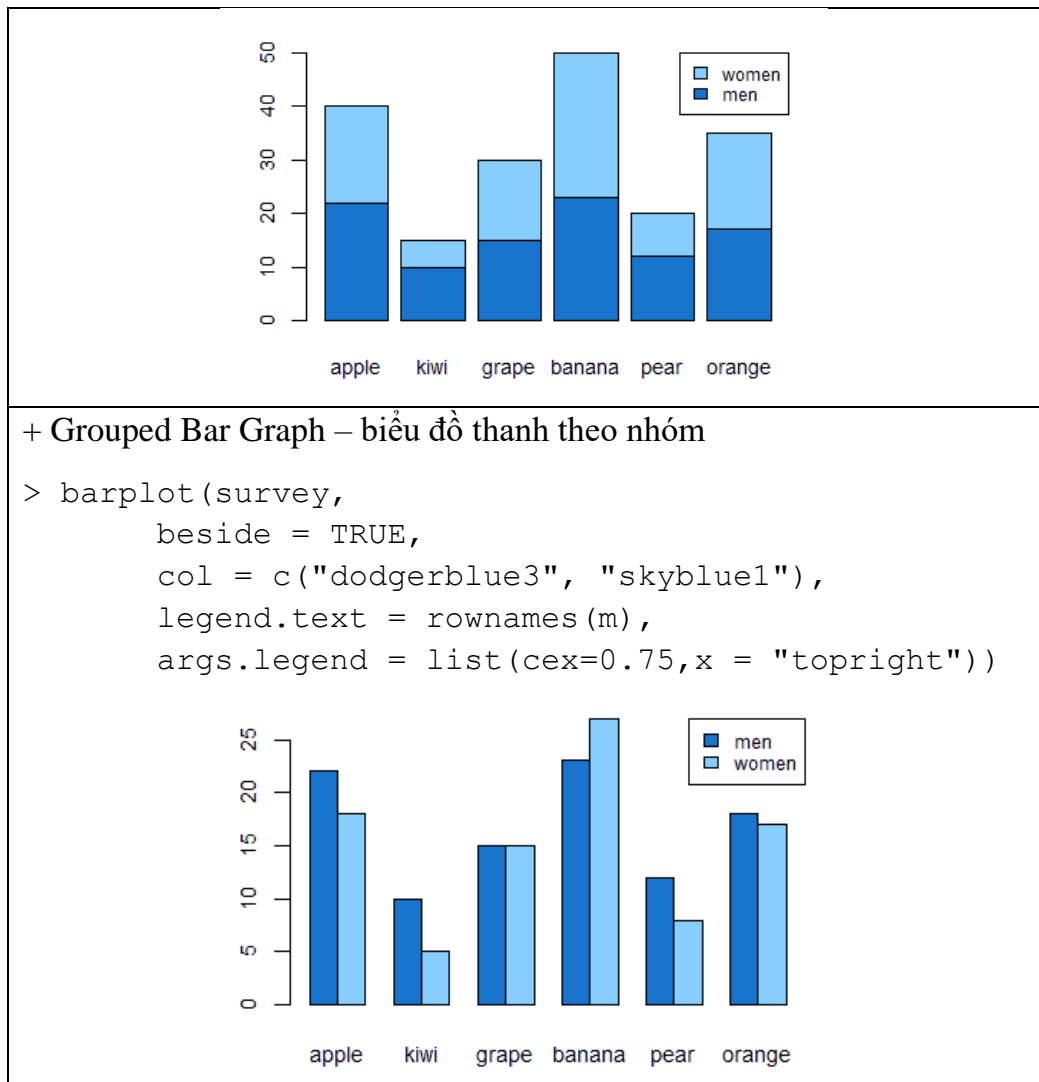
```
> barplot(survey,
col="dodgerblue3",
horiz=TRUE,
las=1)
```



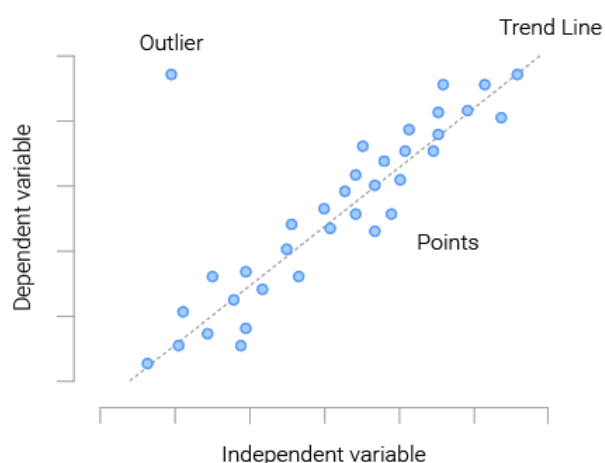
```
> men <- c(apple=22, kiwi=10, grape=15, banana=23,
pear=12, orange=18)
> women <- c(apple=18, kiwi=5, grape=15, banana=27,
pear=8, orange=17)
> survey <- rbind(men, women)
> survey
      apple kiwi grape banana pear orange
men      22   10   15     23    12     18
women    18    5   15     27     8     17
```

+ Stacked Bar Graph – Biểu đồ thanh xếp chồng

```
# Create a stacked barplot and add a legend
> barplot(survey,
col=c("dodgerblue3", "skyblue1"),
legend.text = rownames(m),
args.legend=list(cex=0.75,x = "topright"))
```



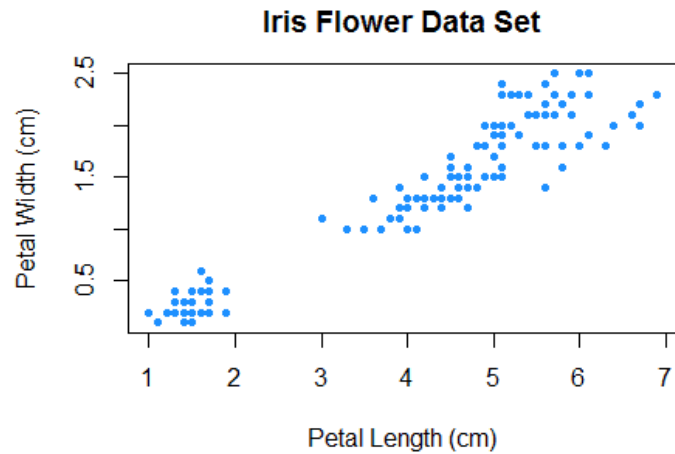
▪ **R Scatter Plot – Biểu đồ phân tán**



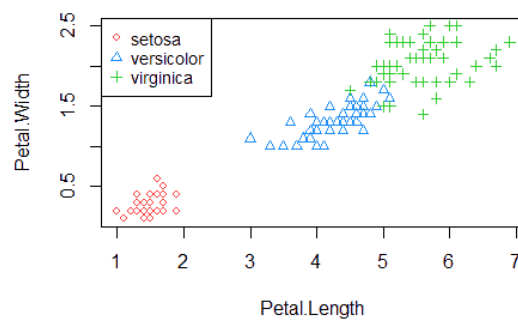
Biểu đồ phân tán là một màn hình đồ họa về mối quan hệ giữa hai tập dữ liệu.

```
plot(Petal.Width ~ Petal.Length, data=iris,
```

```
pch=16,
col="dodgerblue1",
main = "Iris Flower Data Set",
xlab = "Petal Length (cm)",
ylab = "Petal Width (cm)")
```

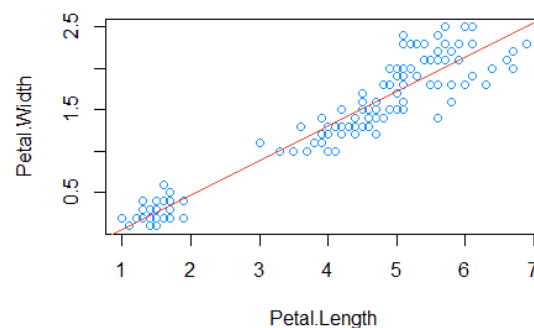


+ Biểu đồ phân tán
với nhiều nhóm



Vẽ đường hồi quy
1. Sử dụng hàm `lm()`
để ước tính mô hình
hồi quy
2. Truyền đối tượng
`lm` vào hàm
`abline()` để vẽ
đường hồi quy

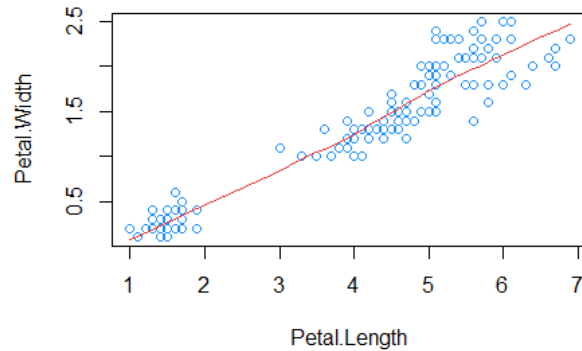
```
> m <- lm(Petal.Width ~ Petal.Length,
data=iris)
> plot(Petal.Width ~ Petal.Length,
data=iris, col="dodgerblue1")
> abline(m, col="brown2")
```



+ Vẽ đường Lowess

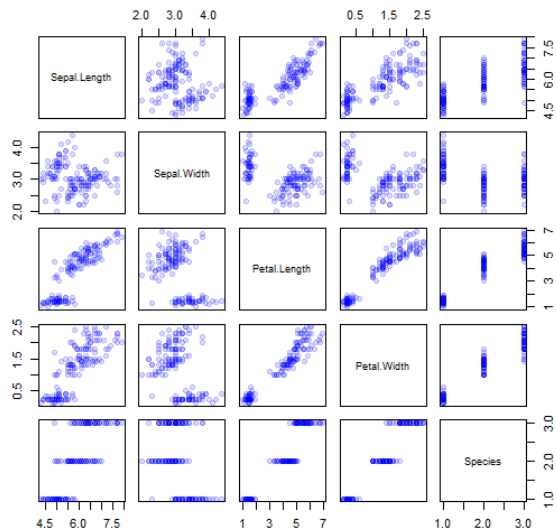
Hàm `lowess()` thực hiện các phép tính để làm mịn đồ thị phân tán có trọng số cục bộ (LOWESS).


```
> plot(Petal.Width ~ Petal.Length, data=iris,
col="dodgerblue1")
> lines(lowess(iris$Petal.Length, iris$Petal.Width), col =
"brown2")
```



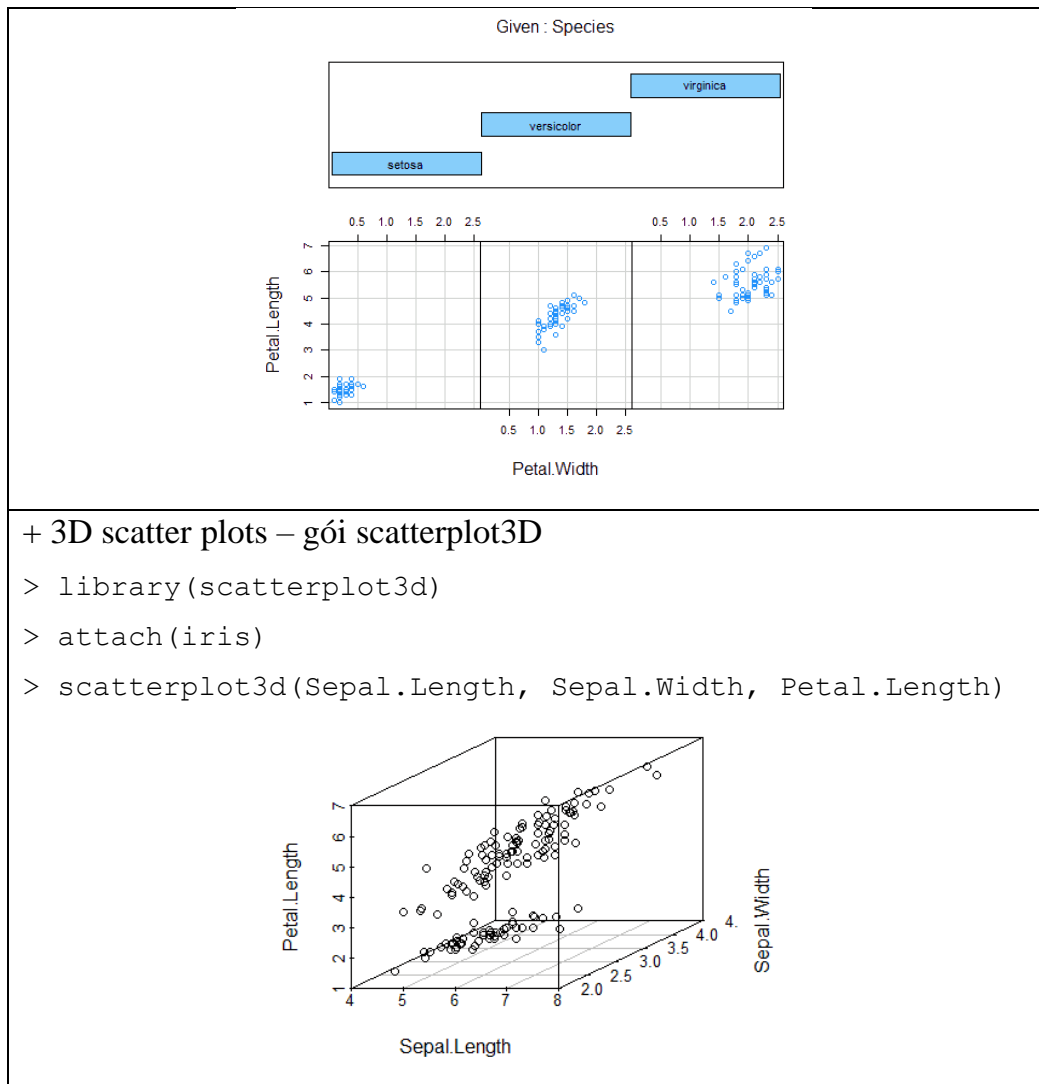
Mã trận đồ thị phân tán

```
> plot(iris,
col=rgb(0,0,1,.15),
pch=19)
```

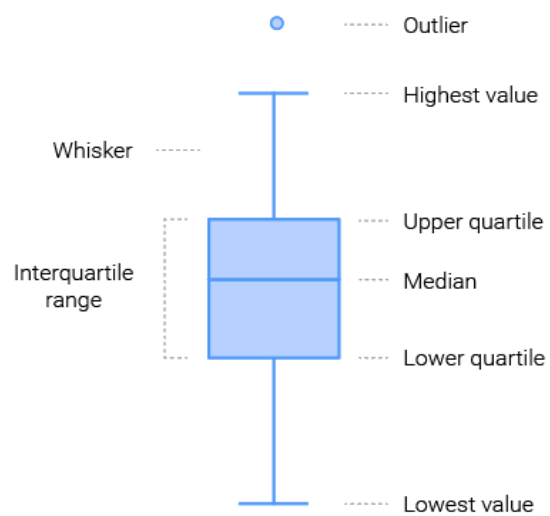


+ Coplots (conditioning scatter plots) – điều kiện của biểu đồ phân tán

```
coplot(Petal.Length ~ Petal.Width | Species,
data=iris,
columns=3,
bar.bg=c(fac="lightskyblue"),
col="dodgerblue1")
```



▪ R Box-whisker Plot – Biểu đồ hộp



Hình 5. 14. Biểu đồ hộp - Box whisker trong R

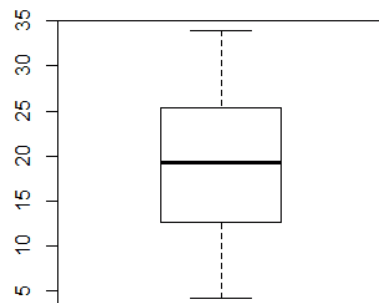
Biểu đồ hình hộp (box-whisker plot hoặc boxplot) là một cách nhanh chóng và dễ dàng để trực quan hóa dữ liệu phức tạp trong đó bạn có nhiều mẫu.

Biểu đồ hộp là một cách tốt để có được bức tranh tổng thể về tập dữ liệu một cách nhỏ gọn.

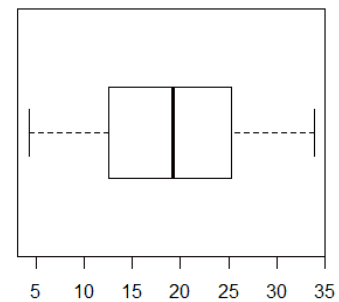
+ Tạo biểu đồ hộp

```
> # First six observations of the 'ToothGrowth' data set
> head(ToothGrowth)
      len supp dose
1   4.2   VC  0.5
2  11.5   VC  0.5
3   7.3   VC  0.5
4   5.8   VC  0.5
5   6.4   VC  0.5
6  10.0   VC  0.5
```

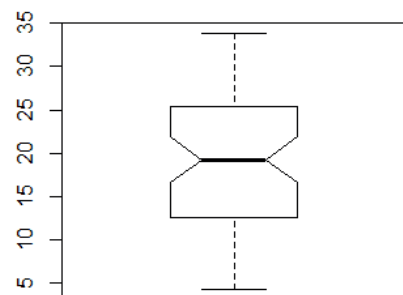
```
> boxplot(ToothGrowth$len)
```



```
> boxplot(ToothGrowth$len,
          horizontal = TRUE)
```

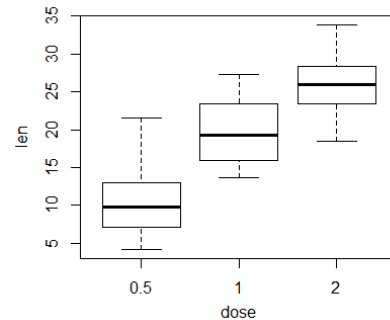


```
> # Add notches to a box
plot
> boxplot(ToothGrowth$len,
          notch = TRUE)
```

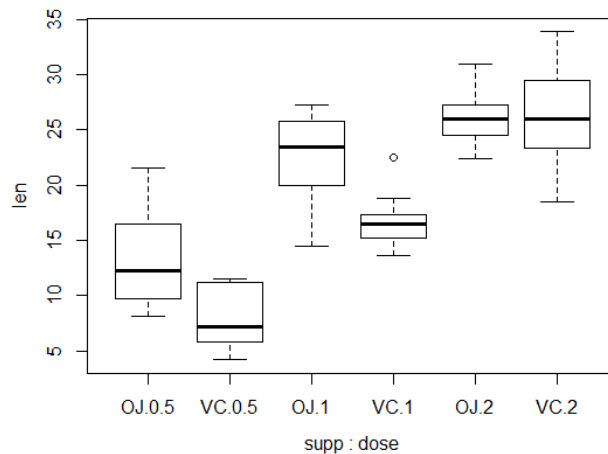


Biểu đồ hình hộp có khía cho phép bạn đánh giá xem các trung bình có khác nhau hay không. Nếu các khía không chồng lên nhau, có bằng chứng chắc chắn (độ tin cậy 95%) các trung tuyến của chúng khác nhau.

```
> # Creating one box plot
for each factor level (dose)
> boxplot(len ~ dose, data =
ToothGrowth)
```

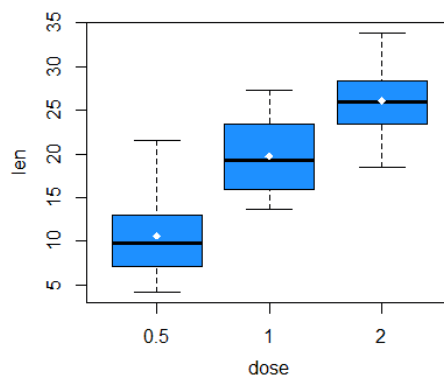


```
> # Box plot of length based on interaction of two
variables (supplement and dose)
> boxplot(len ~ supp*dose, data = ToothGrowth)
```



Biểu đồ hộp được nhóm được sử dụng khi bạn có một biến số, một số nhóm và nhóm con.

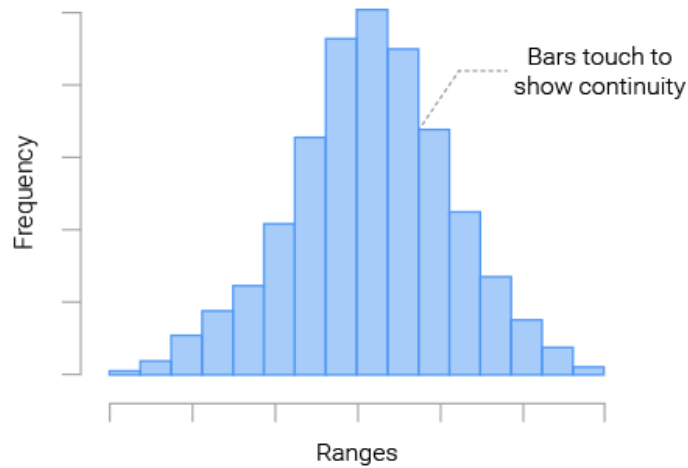
```
> boxplot(len ~ dose, data=ToothGrowth,
col="dodgerblue1")
> meanval <- by(ToothGrowth$len, ToothGrowth$dose, mean)
> points(meanval, col="white", pch=18)
```



Đường ngang ở giữa ô hộp là đường trung bình, không phải giá trị trung bình. Chỉ số trung vị sẽ không giúp bạn hiểu liệu dữ liệu có được phân phối bình thường hay không. Vì vậy, bạn cần thêm các điểm đánh dấu trung bình trên ô hộp của mình.

▪ R Histogram

Biểu đồ Histogram là một hiển thị đồ họa của dữ liệu liên tục bằng cách sử dụng các thanh có độ cao khác nhau.



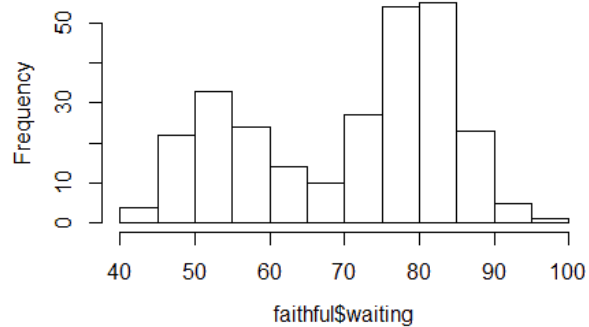
Hình 5. 15. Biểu đồ Histogram trong R

Chúng là một cách tuyệt vời để hiển thị sự phân bố hoặc biến thể của dữ liệu trên một phạm vi.

+ hàm `hist()`

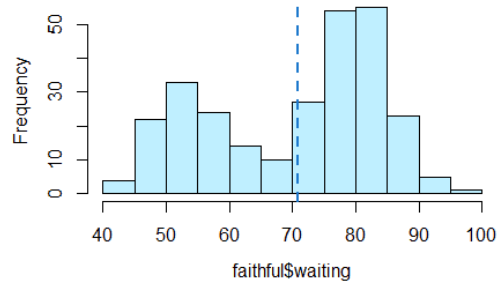
```
> # First six observations of the 'Faithful' data set
> head(faithful)
  eruptions waiting
1    3.600      79
2    1.800      54
3    3.333      74
4    2.283      62
5    4.533      85
6    2.883      55
```

```
> # Create a histogram of time between eruptions of Old
Faithful
> hist(faithful$waiting)
```



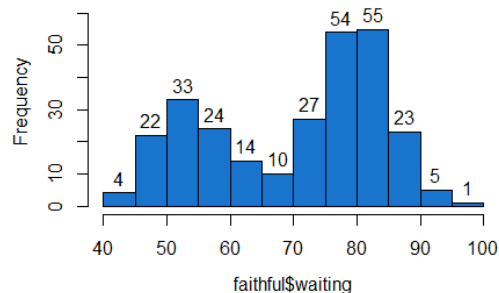
+ Thêm đường trung bình. Sử dụng hàm `abline()`
 + thêm dòng trung bình sẽ cho bạn ý tưởng về mức độ phân phối nằm trên và dưới mức trung bình.

```
> # Add mean line in the histogram
> hist(faithful$waiting,
       col="lightblue1")
> abline(v=mean(faithful$waiting),
       col="dodgerblue3", lty=2, lwd=2)
```



+ thêm giá trị trên đầu thanh với đối số `labels=TRUE`

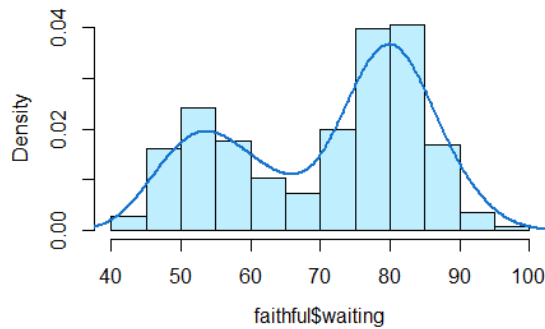
```
> # Show values on top of each bar
in the histogram
> hist(faithful$waiting,
       col="dodgerblue3",
       labels=TRUE)
```



Biểu đồ Kernel Density Estimate (KDE)

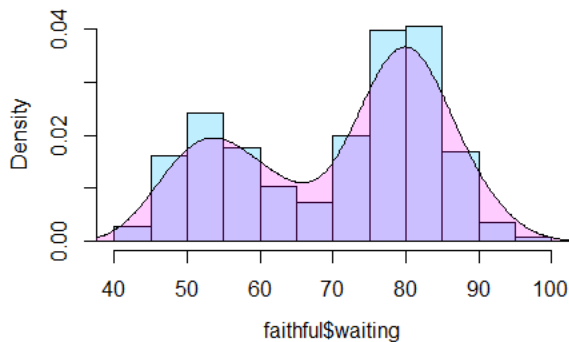
+ Cách hoàn chỉnh nhất để mô tả dữ liệu của bạn là ước tính hàm mật độ xác suất (probability density function - PDF) hoặc mật độ biến của bạn.
 + Sử dụng hàm `density()` để ước lượng mật độ mẫu và sau đó sử dụng hàm `lines()` để vẽ gần đúng.

```
> # Add a kernel density estimate to a histogram
> hist(faithful$waiting,
      col="lightblue1",
      freq = FALSE)
> lines(density(faithful$waiting),
      col="dodgerblue3",
      lwd=2)
```



+ Để lấp đầy biểu đồ mật độ, hãy sử dụng hàm `polygon()`.

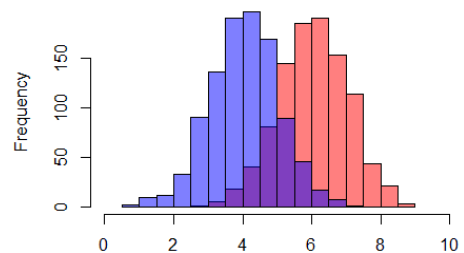
```
> lines(density(faithful$waiting))
> polygon(density(faithful$waiting),
      col=rgb(1,0,1,.2))
```



Vẽ nhiều biểu đồ Histogram

Bạn có thể chồng các biểu đồ bằng cách đặt đối số `add` của biểu đồ thứ hai thành `TRUE`.

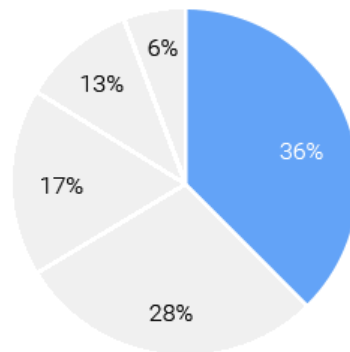
```
> # random numbers
> h1 <- rnorm(1000,6)
> h2 <- rnorm(1000,4)
> # Overlay two
  histograms
> hist(h1,
      col=rgb(1,0,0,0.5))
```



```
> hist(h2,
      col=rgb(0,0,1,0.5),
      add=TRUE)
```

▪ R Pie Chart – biểu đồ tròn

Biểu đồ hình tròn là một biểu đồ đặc biệt hiển thị kích thước tương đối của dữ liệu bằng cách sử dụng các lát hình tròn.



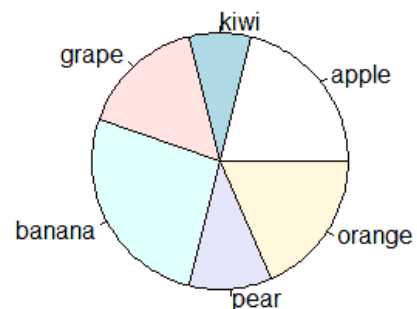
Hình 5. 16. Biểu đồ tròn - Pie trong R

Biểu đồ tròn tốt nếu bạn đang cố gắng so sánh các phần của một chuỗi dữ liệu với toàn bộ.

+ hàm `pie()`

```
> survey <- c(apple=40, kiwi=15, grape=30, banana=50,
  pear=20, orange=35)
> survey
apple kiwi grape banana pear orange
   40   15   30   50   20   35
```

```
> # Create a pie chart
from a vector of data
points
> pie(survey)
```

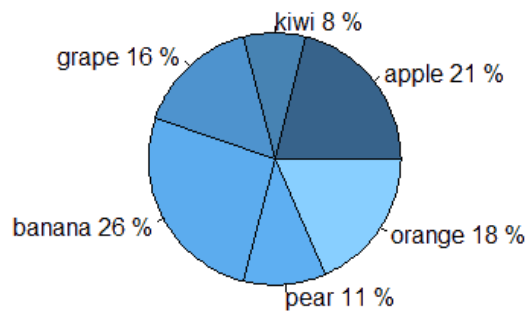


Gắn nhãn biểu đồ hình tròn với tỷ lệ phần trăm


```

> survey <- c(apple=40, kiwi=15, grape=30, banana=50,
pear=20, orange=35)
#calculate percentages
> pct <- round(survey/sum(survey)*100)
#add percents to labels
> lbls <- paste(names(survey), pct, "%")
> pie(survey,
      col=c("steelblue4", "steelblue", "steelblue3",
"steelblue2", "steelblue1", "skyblue1"),
      labels=lbls)

```



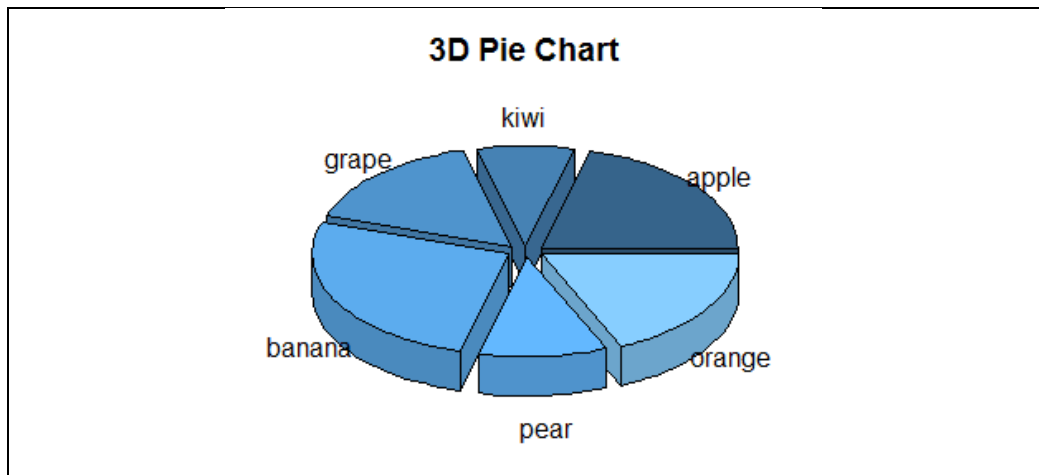
Biểu đồ tròn 3D

+ Để tạo biểu đồ hình tròn 3D, hãy sử dụng hàm `pie3D()` của gói đồ thị và truyền vào vector điểm dữ liệu.

```

> library(plotrix)
> survey <- c(apple=40, kiwi=15, grape=30, banana=50,
pear=20, orange=35)
> pie3D(survey,
        col=c("steelblue4", "steelblue", "steelblue3",
"steelblue2", "steelblue1", "skyblue1"),
        labels = names(survey),
        labelcex = 1,
        explode=0.1,
        theta = 0.8,
        main="3D Pie Chart")

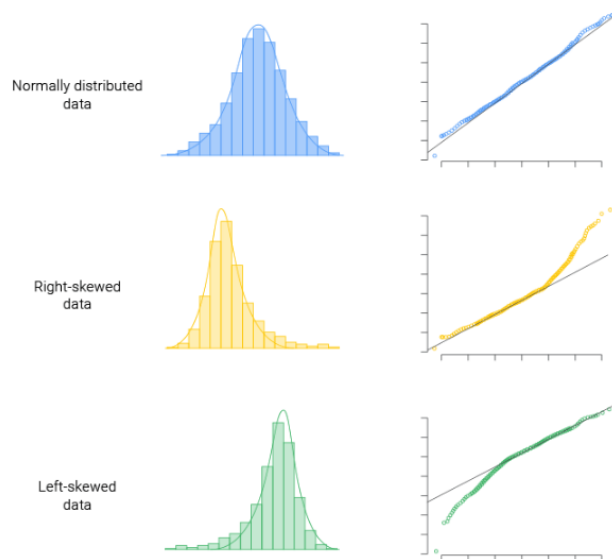
```



▪ R Quantile-Quantile (QQ) Plot

Đôi khi, điều quan trọng là phải biết liệu dữ liệu có được phân phối bình thường hay không. Một biểu đồ lượng tử-lượng tử (biểu đồ Quantile-Quantile - QQ) là một cách kiểm tra đầu tiên tốt.

Nó cho thấy sự phân phối của dữ liệu so với phân phối chuẩn dự kiến.



Nếu dữ liệu được phân phối bình thường, các điểm nằm trên đường tham chiếu 45°. Nếu dữ liệu không bình thường, các điểm sẽ lệch đáng kể so với đường tham chiếu.

Hình 5. 17. Biểu đồ Quantile-Quantile (QQ) trong R

```
> # First six observations of the 'ToothGrowth' data set
> head(ToothGrowth)
  len supp dose
1  4.2   VC  0.5
2 11.5   VC  0.5
```

3	7.3	VC	0.5
4	5.8	VC	0.5
5	6.4	VC	0.5
6	10.0	VC	0.5

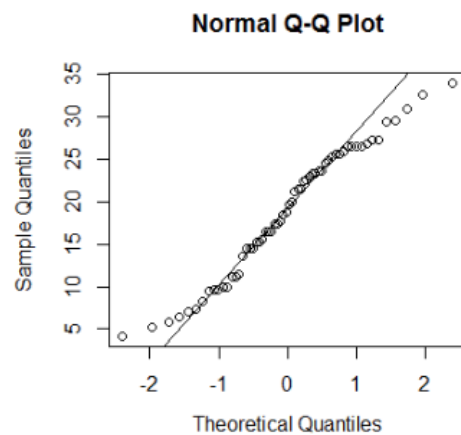
Tạo biểu đồ Normal Quantile-Quantile (QQ) – lượng tử-lượng tử chuẩn

+ hàm `qqnorm()` : vẽ biểu đồ mẫu theo phân phối chuẩn

+hàm `qqline()` : bổ sung một đường phân phối lý thuyết vào biểu đồ QQ chuẩn của bạn. Đường này giúp bạn dễ dàng đánh giá xem các điểm có lệch khỏi đường tham chiếu hay không.

=> Các điểm càng gần đường tham chiếu trong biểu đồ, thì dữ liệu mẫu càng gần với phân phối chuẩn.

```
> # Check 'length'
sample in
'ToothGrowth' dataset
for normality
> with(ToothGrowth, {
  qqnorm(len);
  qqline(len)})
```

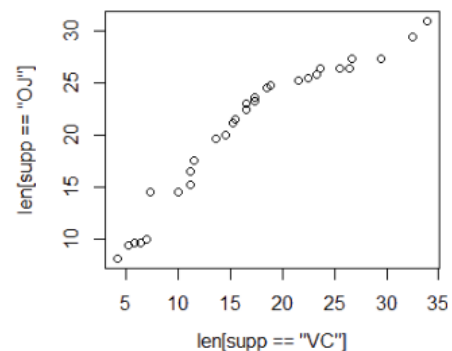


So sánh hai mẫu

+ hàm `qqplot()`

```
> with(ToothGrowth,
  qqplot(len[supp == "VC"],
    len[supp == "OJ"])))
```

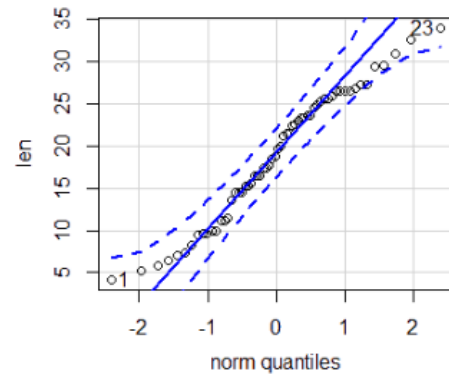
Ví dụ, để kiểm tra xem sự phát triển của răng trong quá trình bổ sung loại axit ascorbic (được mã hóa là VC) và nước cam (mã hóa là OJ) có được phân bố như nhau hay không, bạn chỉ cần thực hiện như sau.



Biểu đồ QQ với khoảng tin cậy

+ Hàm `qqplot()` tiêu chuẩn trong R không cung cấp khoảng tin cậy nhưng hàm `qqPlot()` từ gói `car` thì có.

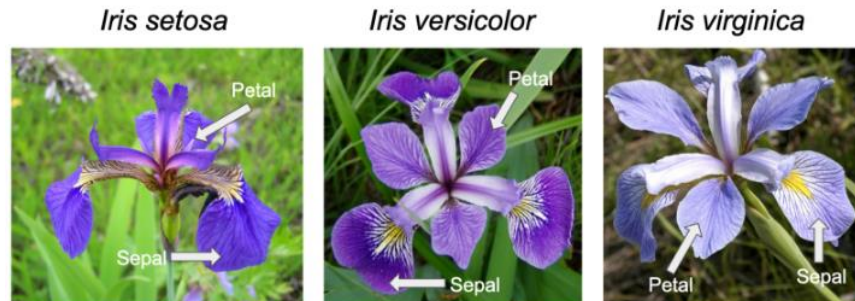
```
> library(car)
> with(ToothGrowth,
      qqPlot(len))
```



VI. Demo chương trình

1. Tập dữ liệu

Iris Dataset : [UCI Machine Learning Repository: Iris Data Set](https://archive.ics.uci.edu/ml/dataset/iris)



Hình 6. 1. Các lớp trong tập dữ liệu Iris

2. Tải dataset lên chương trình

- + Cách 1: Sử dụng file .csv được tải từ liên kết trên
- + Cách 2: Sử dụng Iris dataset trong nền tảng R

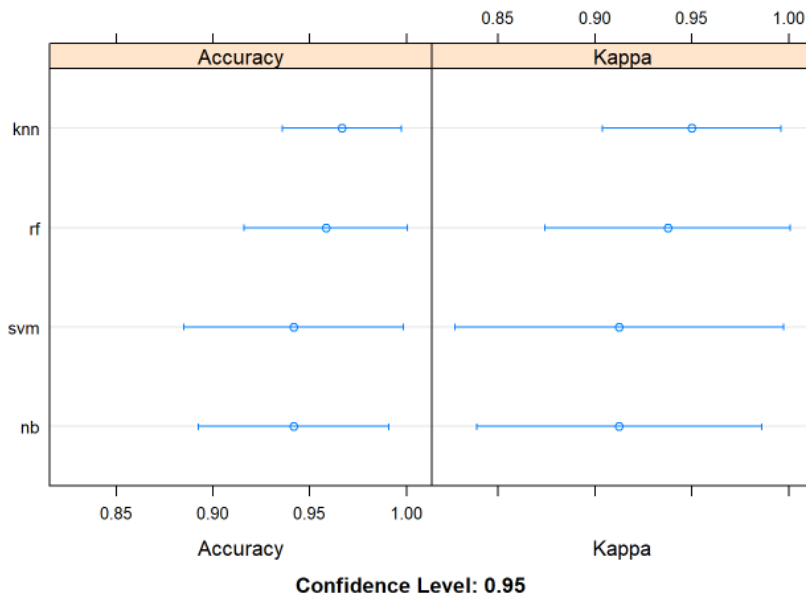
```
> # đính kèm tập dữ liệu iris từ môi trường  
> data(iris)  
> # gắn dữ liệu vào biến dataset  
> dataset <- iris
```

3. Tiền xử lý dữ liệu và xây dựng mô hình

Do các thuật toán randomForest, SVM, KNN, NavieBayes được triển khai trong gói caret, e1071, randomForest, klaR có yêu cầu xử lý dữ liệu và cài đặt khác với thuật toán XgBoost (gói xgboost) nên nhóm chia ra 2 phần để triển khai các thuật toán.

3.1. Các thuật toán sử dụng gói caret

- + Thuật toán randomForest (gói randomforest) có chung cách cài đặt
- + SVM, KNN (gói caret)
- + NavieBayes kết hợp gói caret và klaR
- Chia tập training và validation theo tỷ lệ 80/20
- Sử dụng k-fold với k=10 và độ đo đánh giá (metric) là accuracy
- Kết quả so sánh sau khi chạy các thuật toán. Thuật toán KNN cho các quả tốt nhất.



Hình 6. 2. So sánh kết quả của các thuật toán SVM, KNN, RandomForest, NavieBayes

- Tạo dự đoán từ mô hình tốt nhất (KNN) với tập validation

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  setosa versicolor virginica
## setosa      10         0         0
## versicolor   0         9         0
## virginica    0         1        10
##
## Overall Statistics
##
##           Accuracy : 0.9667
##           95% CI : (0.8278, 0.9992)
##       No Information Rate : 0.3333
##       P-Value [Acc > NIR] : 2.963e-13
##
##           Kappa : 0.95
##
##  McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##           Class: setosa Class: versicolor Class: virginica
## Sensitivity           1.0000           0.9000           1.0000
## Specificity           1.0000           1.0000           0.9500
## Pos Pred Value        1.0000           1.0000           0.9091
## Neg Pred Value        1.0000           0.9524           1.0000
## Prevalence            0.3333           0.3333           0.3333
## Detection Rate        0.3333           0.3000           0.3333
## Detection Prevalence  0.3333           0.3000           0.3667
## Balanced Accuracy     1.0000           0.9500           0.9750
```

Hình 6. 3. Kết quả dự đoán của mô hình KNN trên tập validation

3.2. Thuật toán XgBoost (gói xgboost)

- Chuyển đổi nhãn (labels)

XGBoost yêu cầu các lớp phải ở định dạng số nguyên, bắt đầu bằng 0. Vì vậy, lớp đầu tiên phải là 0. Yếu tố Species được chuyển đổi sang định dạng số nguyên thích hợp.

```
> species = dataset$Species
> label = as.integer(dataset$Species)-1
> dataset$Species = NULL
```

- Chia dữ liệu training và validation tỷ lệ 80/20
- Xác định các tham số chính của mô hình

```
num_class = length(levels(species))
params = list(
  booster="gbtree",
  eta=0.001,
  max_depth=5,
  gamma=3,
  subsample=0.75,
  colsample_bytree=1,
  objective="multi:softprob",
  eval_metric="mlogloss",
  num_class=num_class
)
```

Tham số mục tiêu multi: softprob về cơ bản cung cấp cho chúng ta một phân cụm mờ, trong đó mỗi quan sát được cung cấp một xác suất riêng biệt thuộc về mỗi lớp. Để sử dụng các xác suất này để phân loại, chúng ta sẽ phải xác định xác suất tối đa cho mỗi lần quan sát và ấn định một lớp.

- Kết quả dự đoán trên tập validation

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##           1 10  0  0
##           2  0 10  1
##           3  0  1  8
##
## Overall Statistics
##
##           Accuracy : 0.9333
##           95% CI : (0.7793, 0.9918)
##           No Information Rate : 0.3667
##           P-Value [Acc > NIR] : 1.145e-10
##
##           Kappa : 0.8997
##
## Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##           Class: 1 Class: 2 Class: 3
## Sensitivity           1.0000  0.9091  0.8889
## Specificity           1.0000  0.9474  0.9524
## Pos Pred Value        1.0000  0.9091  0.8889
## Neg Pred Value        1.0000  0.9474  0.9524
## Precision              1.0000  0.9091  0.8889
## Recall                 1.0000  0.9091  0.8889
## F1                     1.0000  0.9091  0.8889
## Prevalence             0.3333  0.3667  0.3000
## Detection Rate         0.3333  0.3333  0.2667
## Detection Prevalence  0.3333  0.3667  0.3000
## Balanced Accuracy      1.0000  0.9282  0.9206
```

Hình 6. 4. Kết quả mô hình XgBoost trên tập validation

VII. Kết luận

Khoa học dữ liệu ngày càng có vai trò quan trọng trong cuộc cách mạng 4.0 với những đầu tàu là dữ liệu lớn ngày càng chứng tỏ tầm quan trọng không thay thế của mình. R là một trong những ngôn ngữ lập trình khoa học dữ liệu, máy học, cũng như các công ty công nghệ hàng đầu thế giới tin tưởng sử dụng trong công việc của họ.

R sở hữu nhiều ưu điểm như miễn phí (mã nguồn mở), kho gói hỗ trợ đồ sộ cho các công việc khai thác, phân tích đánh giá dữ liệu cũng như trong công việc máy học. R trực quan dữ liệu đa dạng và dễ dàng tiếp cận với những người mới bắt đầu, cũng như đã có kinh nghiệm lập trình với các ngôn ngữ lập trình khác. Cú pháp của R đơn giản và cấu trúc dữ liệu của R hỗ trợ tối đa khi làm việc với dữ liệu.

Việc học và biết sử dụng R sẽ mở ra cơ hội việc làm cho các bạn sinh viên cũng như các lập trình viên trong thời buổi dữ liệu lớn nói chung và ngành khoa học dữ liệu nói riêng phát triển một cách việc bậc như hiện nay.

TÀI LIỆU THAM KHẢO

- [1] S. Machlis, ‘Beginner’s guide to R: Syntax quirks you’ll want to know’, *Computerworld*, Feb. 04, 2022.
<https://www.computerworld.com/article/2497319/business-intelligence-beginner-s-guide-to-r-syntax-quirks-you-ll-want-to-know.html> (accessed Mar. 27, 2022).
- [2] CodeLearn, ‘Có Nên Học Ngôn Ngữ Lập Trình R Không ?’
<https://codelearn.io/sharing/co-nen-hoc-ngon-ngu-lap-trinh-r-khong> (accessed Mar. 20, 2022).
- [3] ‘Giới thiệu ngôn ngữ R’, *Viblo*, Oct. 23, 2019. <https://viblo.asia/p/gioi-thieu-ngon-ngu-r-Do754GqJlM6> (accessed Mar. 20, 2022).
- [4] ‘How to Install R on Windows, Mac OS X, and Ubuntu Tutorial’, *DataCamp Community*, Mar. 11, 2020.
<https://www.datacamp.com/community/tutorials/installing-R-windows-mac-ubuntu> (accessed Mar. 20, 2022).
- [5] ‘Learn R By Example’, *Learn By Example*. <https://www.learnbyexample.org/r/> (accessed Mar. 27, 2022).
- [6] ‘Ngôn ngữ lập trình R là gì? – Hành trang Lập trình blog’.
<https://hanhtranglaptrinh.net/ngon-ngu-lap-trinh-r-la-gi/> (accessed Mar. 20, 2022).
- [7] ‘R Language vs Python?’, *Intellipaat Blog*, Aug. 06, 2019.
<https://intellipaat.com/blog/r-vs-python/> (accessed Mar. 20, 2022).
- [8] ‘R Tutorial - Learn R Programming’, *DataMentor*. <https://www.datamentor.io/r-programming/> (accessed Mar. 20, 2022).
- [9] ‘Top 12 R Packages For Machine Learning In 2020’, *Analytics India Magazine*, Jun. 09, 2020. <https://analyticsindiamag.com/top-12-r-packages-for-machine-learning-in-2020/> (accessed Mar. 20, 2022).
- [10] ‘What is R? . MRAN’. <https://mran.microsoft.com/documents/what-is-r> (accessed Mar. 20, 2022).
- [11] J. Brownlee, ‘Your First Machine Learning Project in R Step-By-Step’, *Machine Learning Mastery*, Feb. 02, 2016. <https://machinelearningmastery.com/machine-learning-in-r-step-by-step/> (accessed Mar. 20, 2022).
- [12] ‘Using R for Data Science’, *Master’s in Data Science*.
<https://www.mastersindatascience.org/data-scientist-skills/r/> (accessed Mar. 20, 2022).
- [13] D. Johnson, ‘What is R Programming Language? Introduction & Basics of R’, Jan. 06, 2020. <https://www.guru99.com/r-programming-introduction-basics.html> (accessed Mar. 20, 2022).