Introduction to R

08 Apr 2024
Xian Ji
Introduction to R

08 Apr 2024
Xian JI
What is R

R is a language and environment for statistical computing and graphics.

- Open Source and Free
- Extensive Package Ecosystem
- Statistical Capabilities
- Graphics and Visualization
- ......
R is like a car’s engine while RStudio is like a car’s dashboard

R is a programming language and environment
RStudio is an integrated development environment (IDE)
- **User Interface**: organize files, view plots, and interact with R session
- **Code Editing**: syntax highlighting, code completion, and automatic indentation
- **Integrated Tools**: plotting window, and support for Markdown and R Markdown
- **Customization**: themes, layouts, and keyboard shortcuts
Installing R (Windows)

1. Go to the CRAN website [https://cran.r-project.org/](https://cran.r-project.org/).

2. Click on "Download R for Windows".

3. Click on "install R for the first time" link to download the R executable (.exe) file.

4. Run the R executable file to start installation, and allow the app to make changes to your device.

5. Select the installation language.

6. Follow the installation instructions.

7. Click on "Finish" to exit the installation setup.

8. R has now been successfully installed on your Windows OS.
For MacOS
1. To install R, go to https://cran.r-project.org/.
2. Click Download R for (Mac) OS X.
3. Check the Latest release: section for the appropriate version and follow the directions for download. (Note: Apple silicon vs. Intel chip)

4. Once the file download is complete, click to open the installer. Click Continue and proceed through the installer.
5. Once the R installer has finished, click Close.
Installing RStudio


2. Scroll down to “Installers for Supported Platforms” near the bottom of the page.

3. Click on the download link corresponding to your computer’s operating system.

**IMPORTANT:** RStudio will work only in case if R is already installed.
For MacOS

1. To install RStudio, go to https://www.rstudio.com/products/rstudio/download/
2. Click Download (identify your system)

3. Once the file download is complete, click to open the installer. Drag RStudio into your applications folder.
It has 4 fields:
1. **Source Editor**: script space + data viewer;
2. **Workspace**: R environment + track objects;
3. **Console**: output + executed codes;
4. **Plots/Flies/Helper**: view plots and manage your files, and R help documentation

**R help**
- An invaluable resource for learning about R functions, packages, and syntax.
- Access R help by several ways:
  - Typing `?NAME` or `help(NAME)` at the console, where NAME is the name of the function help is sought for.
  - `?` operator: for a specific function, package, or topic.
  - `?` operator: documentation containing a specific term or keyword.

**Notes:**
- R is case sensitive. ("variable" "variable")
- Commands are separated by a semi-colon (`;`) or by a newline.
- Comments can be put anywhere, starting with a hashmark (`#`); everything to the end of the line is a comment.
- Assign a value to an object by `<-` (assignment operator) or `=` (equals sign).

Example:
```
> x <- 5
> x
[1] 5
> x = 5
> x
[1] 5
```
R help

• An invaluable resource for learning about R functions, packages, and syntax.

• Access R help by several ways:

Typing `?NAME' or `??NAME' or help("NAME") at the console, where NAME is the name of the function help is sought for.

• ? operator: for a specific function, package, or topic
• ?? operator: documentation containing a specific term or keyword.
Notes:

- R is case sensitive. ("Variable" "variable")
- Commands are separated by a semi-colon (;) or by a newline.
- Comments can be put anywhere, starting with a hashmark (#); everything to the end of the line is a comment.
- Assign a value to an object by <- (assignment operator) or = (equals sign).

Example:

```
> x<-5
> x
[1] 5
> x=5
> x
[1] 5
```
R is a programming language and environment.
RStudio is an integrated development environment (IDE)
- **User Interface**: organize files, view plots, and interact with R session
- **Code Editing**: syntax highlighting, code completion, and automatic indentation
- **Integrated Tools**: plotting window, and support for Markdown and R Markdown
- **Customization**: themes, layouts, and keyboard shortcuts
R package

A collection of R functions, data sets, and documentation files bundled together for a specific purpose or task.

1. **Install the package:** This is like installing an app on your phone.

2. **“Load” the package:** “Loading” a package is like opening an app on your phone.

**About update**

**Update R**
1. Visit the CRAN website: [https://cran.r-project.org/](https://cran.r-project.org/)
2. Download the latest version of R for Windows.
3. Run the downloaded installer and follow the installation instructions.
4. Make sure to choose the option to update existing installation when prompted.
   Tools → global option

**Update RStudio**
1. Go to Help → Check for Updates
2. If an update is available, follow the prompts to download and install the latest version

**Update R packages**
1. Click on Packages → Manage Packages
2. Follow the prompts to update all packages or an individual package.
install (download) an extra package which is not yet available on your system

1. Using R code to install the package
   ```r
   install.packages("package")
   ```

2. Using RStudio IDE
   Go to Tools -> Install Packages.
   Type package name and click Install.

3. Install from local file (.zip; .tar.gz)
   ```r
   install.packages("/path/to/package/package_name.tar.gz", repos = NULL, type = "source")
   ```

4. Use “Packages” window to install
   In the “Packages” window, click on “Install Packages” button:
   ![Image of the "Packages" window]

   Then, complete the dialog window as follows and click “Install”:
   ![Image of the dialog window]

5. Using devtools package: If the package is hosted on GitHub or other version control repositories
   ```r
   install.packages("devtools")
   library(devtools)
   install_github("cran/BayesTree")
   ```

Note: **Uninstall** packages

1. remove.packages("package-to-remove")
   e.g. remove.packages("BayesTree")
load the installed package BayesTree:

library(BayesTree)

Tips:

✓ To find out which additional packages are installed on your system, type in console 3 library():

Remark:
make it clear what package an object comes from use the package name followed by the name of the object, like dplyr::filter()

dplyr is the name of the package, filter is the name of a function in that package.
About update

Update R
1. Visit the CRAN website: https://cran.r-project.org/
2. Download the latest version of R for Windows.
3. Run the downloaded installer and follow the installation instructions.
4. Make sure to choose the option to update existing installation when prompted.
   Tools -> global option

Update RStudio
1. Go to Help -> Check for Updates
2. If an update is available, follow the prompts to download and install the latest version

Update R packages
1. list all packages where an update is available
   install.packages()
2. update all available packages
   update.packages()
3. update, without prompts for permission/clarification
   update.packages(all = FALSE)
4. update only a specific package use install.packages()
   install.packages("NAME")
What is R

R is a language and environment for statistical computing and graphics.

- Open Source and Free
- Extensive Package Ecosystem
- Statistical Capabilities
- Graphics and Visualization
- ...
Data Structure & Im/Exporting Data

### Data type

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The most primitive form of data, consisting of characters and can be manipulated as a single entity.</td>
<td>“Hello”</td>
</tr>
<tr>
<td>Integer</td>
<td>Represents whole numbers (positive or negative) and can be used for counting or indexing.</td>
<td>100</td>
</tr>
<tr>
<td>Float</td>
<td>Represents real numbers with a fractional part.</td>
<td>3.14</td>
</tr>
<tr>
<td>Boolean</td>
<td>Represents true or false values, which can be used for logical operations.</td>
<td>True</td>
</tr>
<tr>
<td>Complex</td>
<td>A data type representing numbers with both real and imaginary parts.</td>
<td>2 + 3i</td>
</tr>
</tbody>
</table>

### Converting data type

- **Convert string to integer**: Converts a string to an integer (e.g., “123” to 123).
- **Convert integer to string**: Converts an integer to a string (e.g., 123 to “123”).
- **Convert boolean to integer**: Converts a boolean value to an integer (e.g., True to 1, False to 0).
- **Convert float to integer**: Converts a float to an integer (e.g., 3.14 to 3).

### Data Structures

- **Array**: A collection of elements of the same type stored in contiguous memory locations.
- **List**: An ordered collection of elements, often implemented as a linked list.
- **Dictionary**: A collection of key-value pairs, allowing fast retrieval by key.
- **Set**: An unordered collection of unique elements, often implemented as a hash table.
- **Tuple**: An ordered collection of elements, similar to an array but immutable.

### Basic Functions

- **Create your own function**
  1. Define the function with a signature that reflects its purpose.
  2. Implement the logic within the function body.
  3. Test the function with different inputs to ensure correctness.

### Import/Export data

- **Good Format**
  - CSV: Comma-separated values file, commonly used for data exchange.
  - JSON: JavaScript Object Notation, a lightweight data interchange format.
  - SQLite: A database management system that efficiently stores and retrieves data.

- **Bad Format**
  - Unstructured files: Files without a clear format or structure, making data extraction difficult.
  - Binary files: Files that are not easily parsed by humans or software.

- **Import/Export Tools**
  - Skype: A platform for real-time communication, used for data sharing.
  - Python libraries (like pandas): Powerful tools for data manipulation and analysis.
# Data type

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>This data type represents numerical values, including integers and floating-point numbers.</td>
<td>5, -3.14</td>
</tr>
<tr>
<td>Character</td>
<td>Character data type represents text or strings.</td>
<td>&quot;hello&quot;, 'R programming'</td>
</tr>
<tr>
<td>Integer</td>
<td>Integer data type represents whole numbers without fractional parts.</td>
<td>10, -100</td>
</tr>
<tr>
<td>Logical</td>
<td>Logical data type represents boolean values, which can be either TRUE or FALSE.</td>
<td>TRUE, FALSE</td>
</tr>
<tr>
<td>Factor</td>
<td>Factor data type represents categorical variables with a fixed set of levels or categories.</td>
<td>factor(c(&quot;male&quot;, &quot;female&quot;))</td>
</tr>
<tr>
<td>Complex</td>
<td>Complex data type represents complex numbers with real and imaginary parts.</td>
<td>3 + 4i, -2 - 3i</td>
</tr>
</tbody>
</table>

Note: Using class() to identify the data type

### Converting data type

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>as.character()</td>
<td>Converts an object to a character (string) type.</td>
</tr>
<tr>
<td>as.numeric()</td>
<td>Converts an object to a numeric (numeric) type.</td>
</tr>
<tr>
<td>as.integer()</td>
<td>Converts an object to an integer (integer) type.</td>
</tr>
<tr>
<td>as.logical()</td>
<td>Converts an object to a logical (boolean) type.</td>
</tr>
<tr>
<td>as.factor()</td>
<td>Converts an object to a factor type.</td>
</tr>
</tbody>
</table>

Note: Any numeric value which is not 0, on conversion to Logical type gets converted to TRUE.
# Data Structures

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Mode (data “type”)</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vector</strong></td>
<td>1</td>
<td>Identical</td>
</tr>
<tr>
<td></td>
<td>m…</td>
<td></td>
</tr>
<tr>
<td><strong>Matrix</strong></td>
<td>n</td>
<td>Identical</td>
</tr>
<tr>
<td></td>
<td>m…</td>
<td></td>
</tr>
<tr>
<td><strong>List</strong></td>
<td></td>
<td>Can be different</td>
</tr>
<tr>
<td><strong>Data frame</strong></td>
<td>n</td>
<td>Can be different</td>
</tr>
</tbody>
</table>

### Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>c()</code></td>
<td>Concatenates or combines multiple vectors into a single vector.</td>
</tr>
<tr>
<td><code>append()</code></td>
<td>Appends elements to a vector.</td>
</tr>
<tr>
<td><code>rev()</code></td>
<td>Reverses the order of elements in a vector.</td>
</tr>
<tr>
<td><code>sort()</code></td>
<td>Sorts the elements of a vector in ascending order.</td>
</tr>
<tr>
<td><code>order()</code></td>
<td>Returns the index that would sort a vector.</td>
</tr>
<tr>
<td><code>unique()</code></td>
<td>Removes duplicates from a vector.</td>
</tr>
<tr>
<td><code>rep()</code></td>
<td>Repeats elements of a vector.</td>
</tr>
<tr>
<td><code>matrix()</code></td>
<td>Creates a matrix.</td>
</tr>
<tr>
<td><code>diag()</code></td>
<td>Creates a diagonal matrix from a vector.</td>
</tr>
<tr>
<td><code>dim()</code></td>
<td>Gets or sets the dimensions of a matrix.</td>
</tr>
<tr>
<td><code>rownames()</code></td>
<td>Gets or sets row names of a matrix.</td>
</tr>
<tr>
<td><code>colnames()</code></td>
<td>Gets or sets column names of a matrix.</td>
</tr>
<tr>
<td><code>t()</code></td>
<td>Transposes a matrix.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>list()</code></td>
<td>Creates a list.</td>
</tr>
<tr>
<td><code>unlist()</code></td>
<td>Converts a list to a vector.</td>
</tr>
<tr>
<td><code>append()</code></td>
<td>Appends elements to a list.</td>
</tr>
<tr>
<td><code>names()</code></td>
<td>Gets or sets names of list elements.</td>
</tr>
<tr>
<td><code>length()</code></td>
<td>Determines the length or number of elements in a data structure.</td>
</tr>
<tr>
<td><code>unname()</code></td>
<td>Removes names from an object, making it unnamed.</td>
</tr>
<tr>
<td><code>is()</code></td>
<td>A family of functions such as <code>is.vector()</code>, <code>is.matrix()</code>, <code>is.list()</code>, etc., to check if an object belongs to a particular class or data structure type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>head()</code></td>
<td>Displays the first few rows of the data frame to quickly inspect the data.</td>
</tr>
<tr>
<td><code>tail()</code></td>
<td>Displays the last few rows of the data frame.</td>
</tr>
<tr>
<td><code>str()</code></td>
<td>Provides a concise summary of the structure of the data frame, including data types of each column.</td>
</tr>
<tr>
<td><code>summary()</code></td>
<td>Generates a summary of each variable in the data frame, including measures like mean, median, etc.</td>
</tr>
<tr>
<td><code>dim()</code></td>
<td>Returns the dimensions (number of rows and columns) of the data frame.</td>
</tr>
<tr>
<td><code>names()</code></td>
<td>Returns the names of the columns in the data frame.</td>
</tr>
<tr>
<td><code>colnames()</code></td>
<td>Returns the names of the columns in the data frame.</td>
</tr>
<tr>
<td><code>rownames()</code></td>
<td>Returns the names of the rows in the data frame.</td>
</tr>
<tr>
<td><code>nrow()</code></td>
<td>Returns the number of rows in the data frame.</td>
</tr>
<tr>
<td><code>ncol()</code></td>
<td>Returns the number of columns in the data frame.</td>
</tr>
<tr>
<td><code>row()</code></td>
<td>Returns the number of rows in the data frame.</td>
</tr>
</tbody>
</table>
Basic Functions

<table>
<thead>
<tr>
<th>Arithmetic Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+, -, *, /</td>
<td>Addition, subtraction, multiplication, and division.</td>
</tr>
<tr>
<td>^ or **</td>
<td>Exponentiation.</td>
</tr>
<tr>
<td>sqrt()</td>
<td>Square root.</td>
</tr>
<tr>
<td>abs()</td>
<td>Absolute value.</td>
</tr>
<tr>
<td>round()</td>
<td>Round to the nearest integer.</td>
</tr>
<tr>
<td>% %</td>
<td>Modulo (remainder of division).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean()</td>
<td>Calculates the mean (average) of the data.</td>
</tr>
<tr>
<td>median()</td>
<td>Computes the median (middle value) of the data.</td>
</tr>
<tr>
<td>sd()</td>
<td>Calculates the standard deviation of the data.</td>
</tr>
<tr>
<td>var()</td>
<td>Computes the variance of the data.</td>
</tr>
<tr>
<td>max()</td>
<td>Finds the maximum value in the data.</td>
</tr>
<tr>
<td>min()</td>
<td>Finds the minimum value in the data.</td>
</tr>
<tr>
<td>quantile()</td>
<td>Calculates quantiles of the data.</td>
</tr>
<tr>
<td>summary()</td>
<td>Provides summary statistics of the data.</td>
</tr>
</tbody>
</table>

Create your own function

1. **Define Your Function**: Use the `function()` keyword to start defining your function.

2. **Provide Arguments**: Specify what information your function will take as input.

3. **Write the Function Body**: Inside curly braces `{}`, write the code that tells your function what to do.

4. **Return Values (Optional)**: Use `return()` to specify what your function should output.

5. **Assign Your Function a Name**: Give your function a name so you can use it later in your code.

Exercise: Write a function Pythagoras to calculate the length of hypotenuse from length of legs in right-angled triangle (Theorem of Pythagoras)

\[ c^2 = a^2 + b^2 \]
Import/ Export data

**IMPORT**
Environment > Import Dataset > From Text...

or
read.csv("your.path/filename.csv", header=TRUE) CSV
read.table("file.txt", header = TRUE, sep = "\t") TXT

or
read.csv(file.choose()) CSV
read.table(file.choose()) TXT

**EXPORT**
write.csv(df, "Path/FileName.csv", row.names=FALSE)
write.table(df, file = "Path/FileName.txt", sep = "\t", row.names = FALSE)

**Why Import Database into R?**
- Efficiency: Analyze large datasets without loading them entirely into memory.
- Flexibility: Process and analyze data directly from databases.
- Integration: Seamlessly integrate database operations with R’s analytical capabilities.

**SQL**
Database

- Avoid names with special symbols: ?, $, %, @, ^, (, ), {, }, [...]. Only underscore can be used.
- Avoid beginning variable names with a number. Use letter instead. Good column names: sport_100m or x100m. Bad column name: 100m
- Column names must be unique. Duplicated names are not allowed.
- R is case sensitive. This means that Name is different from Name or NAME.
- Avoid blank rows in your data
- Delete any comments in your file
- Replace missing values by NA (for not available)
- If you have a column containing date, use the four digit format. Good format: 01/01/2016. Bad format: 01/01/16

**IMPORT**
Environment > Import Dataset > From Excel...

or
library(openxlsx)
data <- read.xlsx("path/to/your/file.xlsx")

**EXPORT**
data, "path/to/your/output.xlsx")
IMPORT
Environment > Import Dataset > From Text ...

or
read.csv("your.path/filename.csv", header=TRUE) CSV
read.table("file.txt", header = TRUE, sep = "\t") TXT
or
read.csv(file.choose()) CSV
read.table(file.choose()) TXT

EXPORT
write.csv(df, "Path/File Name.csv", row.names=FALSE)
write.table(df, file = "Path/File Name.txt", sep = "\t", row.names = FALSE)
- **Avoid names with blank spaces.** Good column names: `Long_jump` or `Long_jump`. Bad column name: `Long jump`.
- **Avoid names with special symbols:** ?, $, *, +, #, (, ), ~, /, }, {}, |, >, < etc. Only underscore can be used.
- **Avoid beginning variable names with a number.** Use letter instead. Good column names: `sport_100m` or `x100m`. Bad column name: `100m`.
- **Column names must be unique.** Duplicated names are not allowed.
- **R is case sensitive.** This means that Name is different from Name or NAME.
- **Avoid blank rows in your data**
- **Delete any comments in your file**
- **Replace missing values by NA (for not available)**
- **If you have a column containing date, use the four digit format.** Good format: 01/01/2023. Bad format: 01/01/16
IMPORT
Environment > Import Dataset > From Excel...

or

library(openxlsx)
data <- read.xlsx("path/to/your/file.xlsx")

EXPORT
write.xlsx(data, "path/to/your/output.xlsx")
Why Import Database into R?

**Efficiency**: Analyze large datasets without loading them entirely into memory.
**Scalability**: Process and analyze data directly from databases.
**Integration**: Seamlessly integrate database operations with R’s analytical capabilities.

The **DBI (Database Interface) package** provides a simple, consistent interface between R and database management systems (DBMS).

R Packages: dbplyr, DBI, RSQLite, RMySQL, RPostgreSQL, etc.
Supported Databases: SQLite, MySQL/MariaDB, PostgreSQL, SQL Server, Oracle, etc.

```
library(DBI)
lapply(RSQLite)

# connect to the database
con <- dbConnect(SQLLite, "path/to/your/database.sqlite.db")

# Reading the data from database
df <- dbReadTable(con, "yourtablename")
df <- dbGetQuery(con, "SQL query")

# saving data to the database
dbWriteTable(con, "tablename", table)

Close the connection
dbDisconnect(con)
```

**NOTE**: Closing the Connection
It’s important to remember to close the connection using the `dbDisconnect()` function when you’re done working with the database to release resources and avoid potential memory leaks.