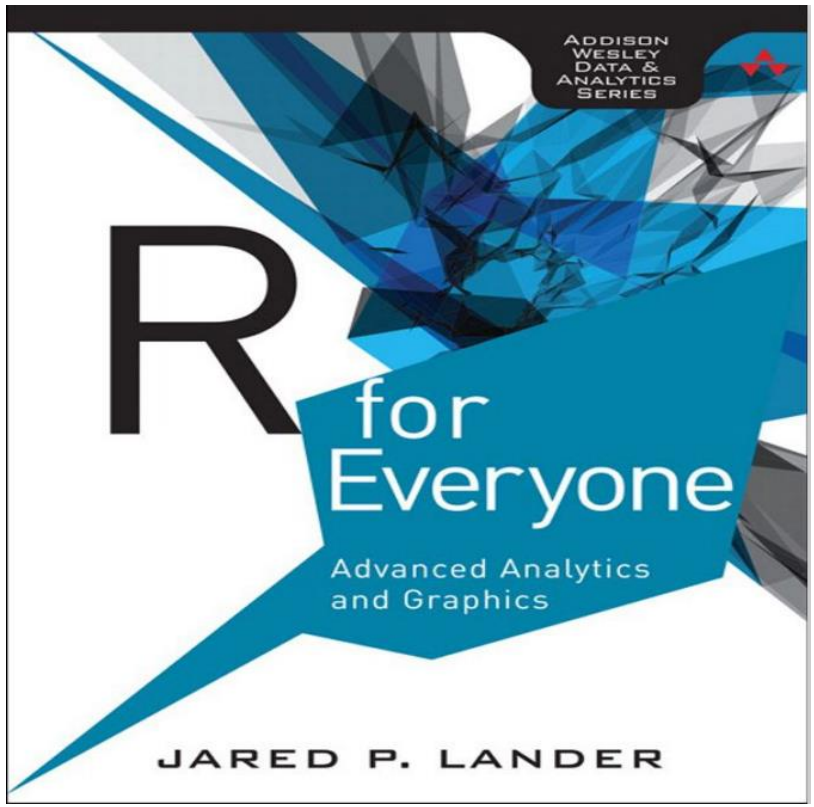
Introduction to R

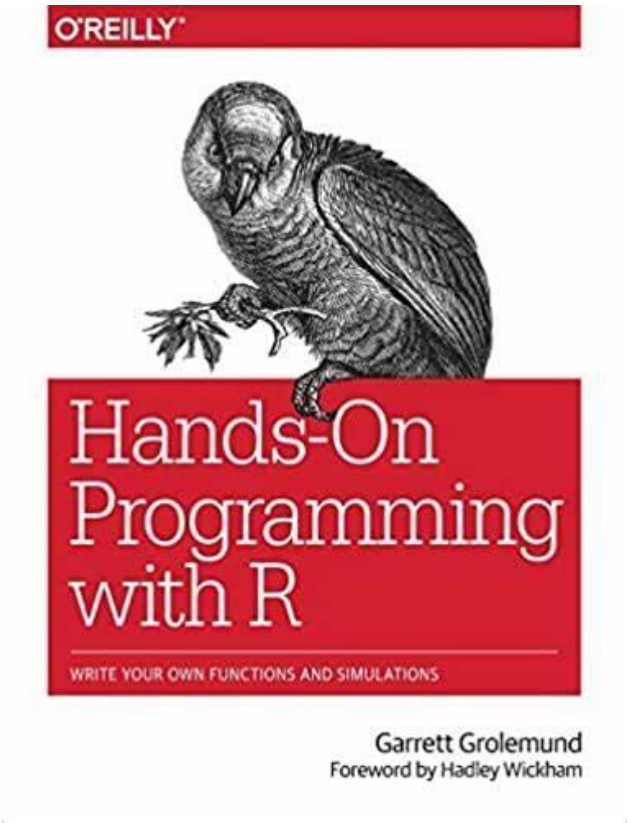
Panchatcharam M

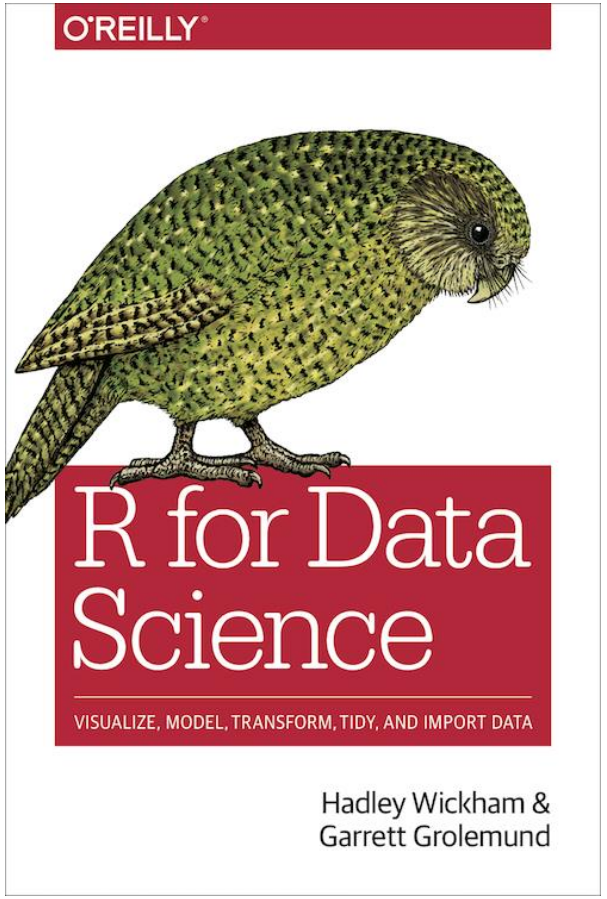
Associate Professor

Department of Mathematics and Statistics,
IIT Tirupati

Reference


Addison Wesley Data & Analytics Series
R for Everyone
Advanced Analytics and Graphics
JARED P. LANDER


O'REILLY
Hands-On Programming with R
WRITE YOUR OWN FUNCTIONS AND SIMULATIONS
Garrett Grolemund
Foreword by Hadley Wickham


O'REILLY
R for Data Science
VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA
Hadley Wickham &
Garrett Grolemund

What, Why, History

What is R?

- Popular Programming Language
- Used for Statistical Computing
- Graphical Representation
- Analyze and Visualize Data

Why R?

- Great Resource for Data Analysis
- Good for Data Visualization
- Widely used for Data Science
- Commonly used in Machine Learning
- Open Source and Fee

- It has a plenty of statistical techniques
  - Statistical Tests
  - Classification
  - Clustering
  - Data Reduction
- It has many packages or libraries

- It is easy to draw many graphs
  - Pie Chart
  - Histogram
  - Box plot
  - And so on

- Lightweight: Excel and SPSS
- Heavy Duty: High-performance analysis built with C++
- Personal Computer to Advance Business Process
  - R, Python, etc

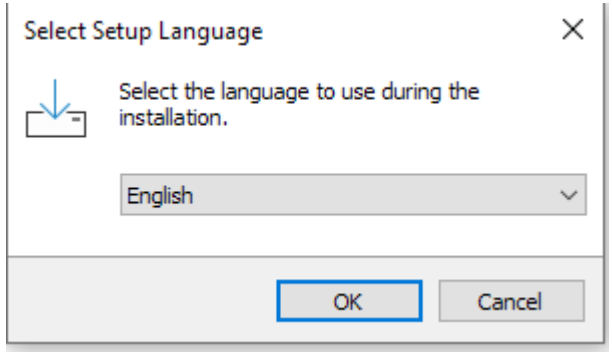
- Invented by Robert Gentleman and Ross Ihaka, University of Auckland, 1993
- Grew out of S which was invented by John Chambers at Bell Labs
- R became popular since 2000's because of its usage in banking, marketing, pharma, politics, genomics et

R INSTALLATION ON WINDOWS

Downloading R

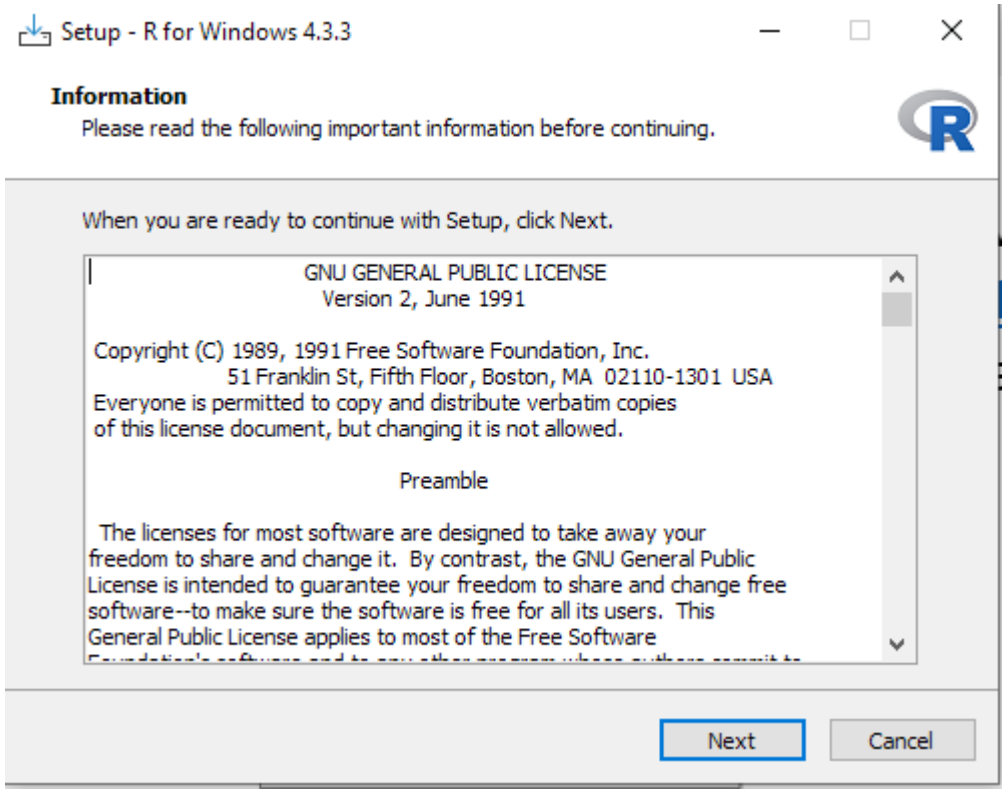
- Visit the following website
- The Comprehensive R Archive Network (r-project.org)
- https://cran.r-project.org/
- Depending on the OS, download the respective file

- Visit the following website and download the windows binaries
- https://cran.r-project.org/bin/windows/base/R-4.3.3-win.exe
- Go to Downloads (or where the exe is downloaded)
- Double click the R-4.3.3-win.exe file

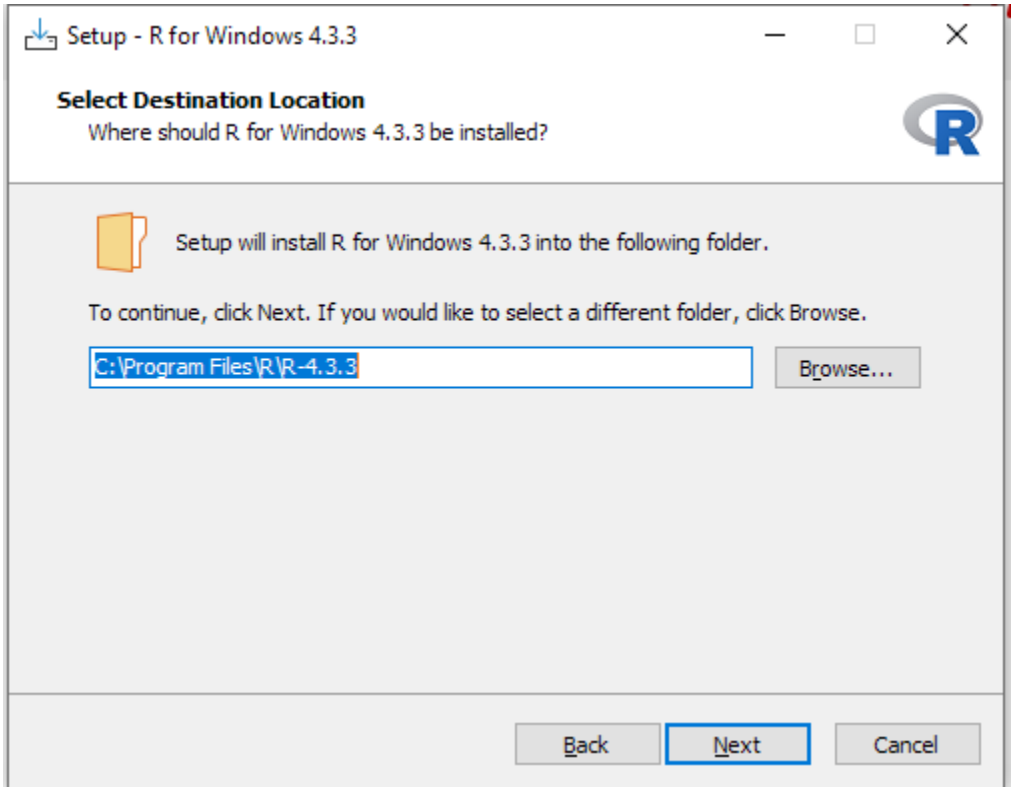


- Click OK on the next dialog box

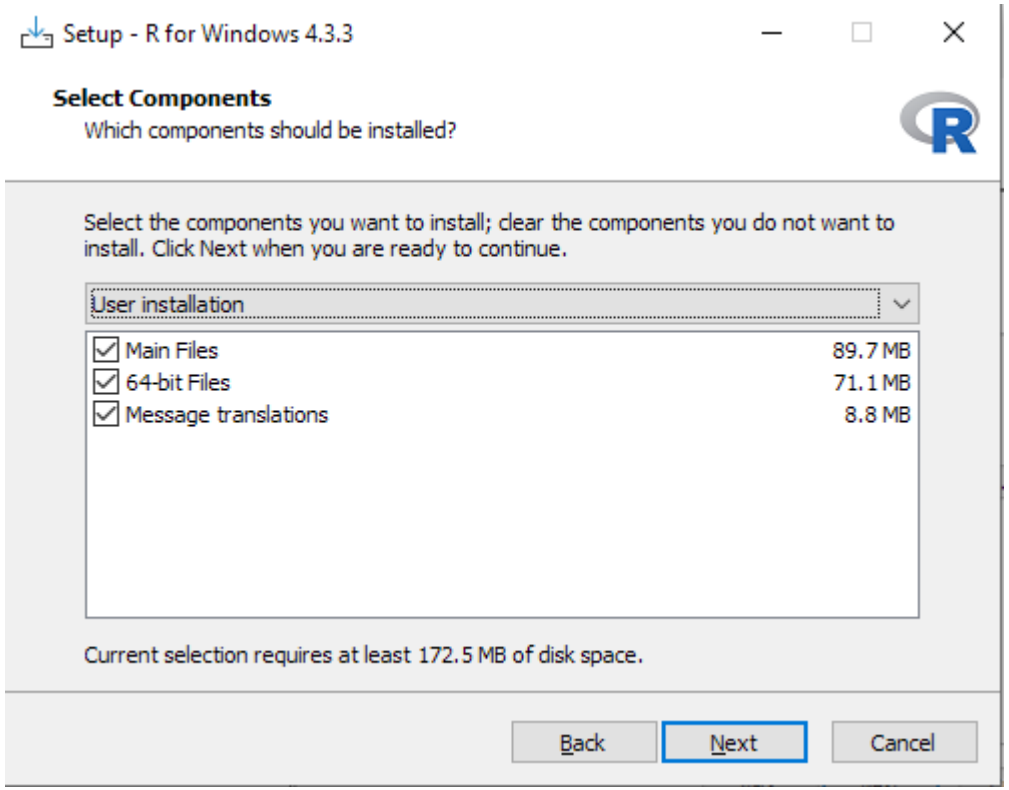
- Click Next


Setup - R for Windows 4.3.3
Information
Please read the following important information before continuing.
When you are ready to continue with Setup, click Next.
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users. This
General Public License applies to most of the Free Software
Next
Cancel

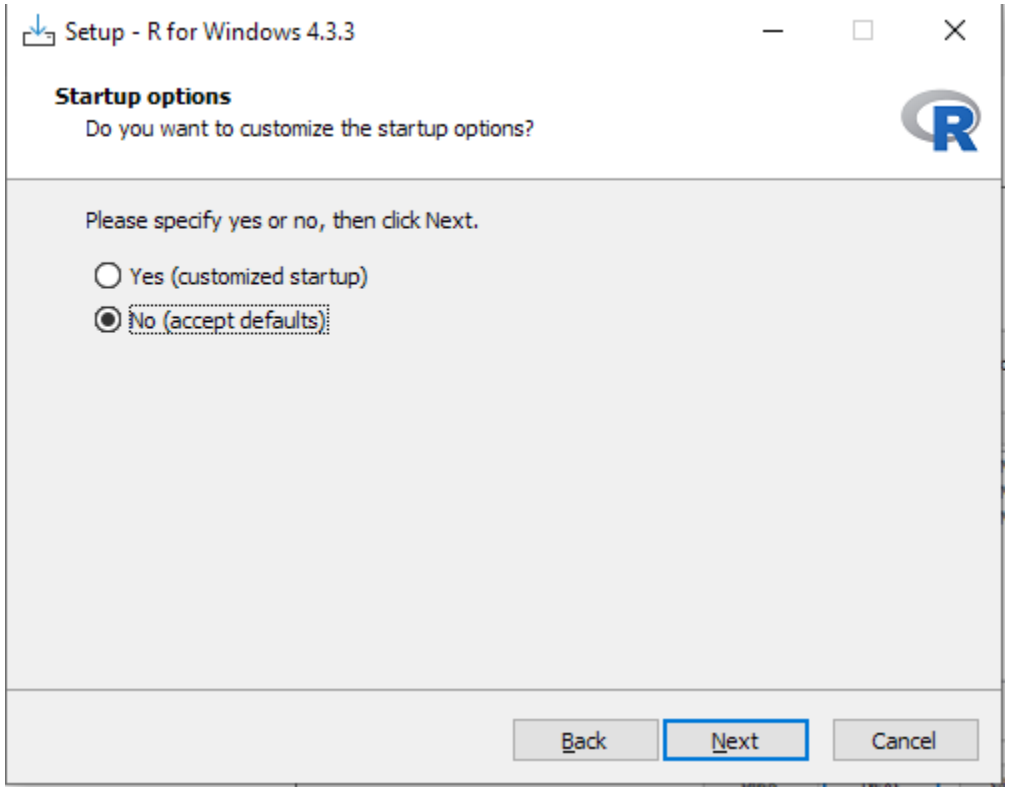
- Click Next


Setup - R for Windows 4.3.3
Select Destination Location
Where should R for Windows 4.3.3 be installed?
Setup will install R for Windows 4.3.3 into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files\R\R-4.3.3
Browse...
Back
Next
Cancel

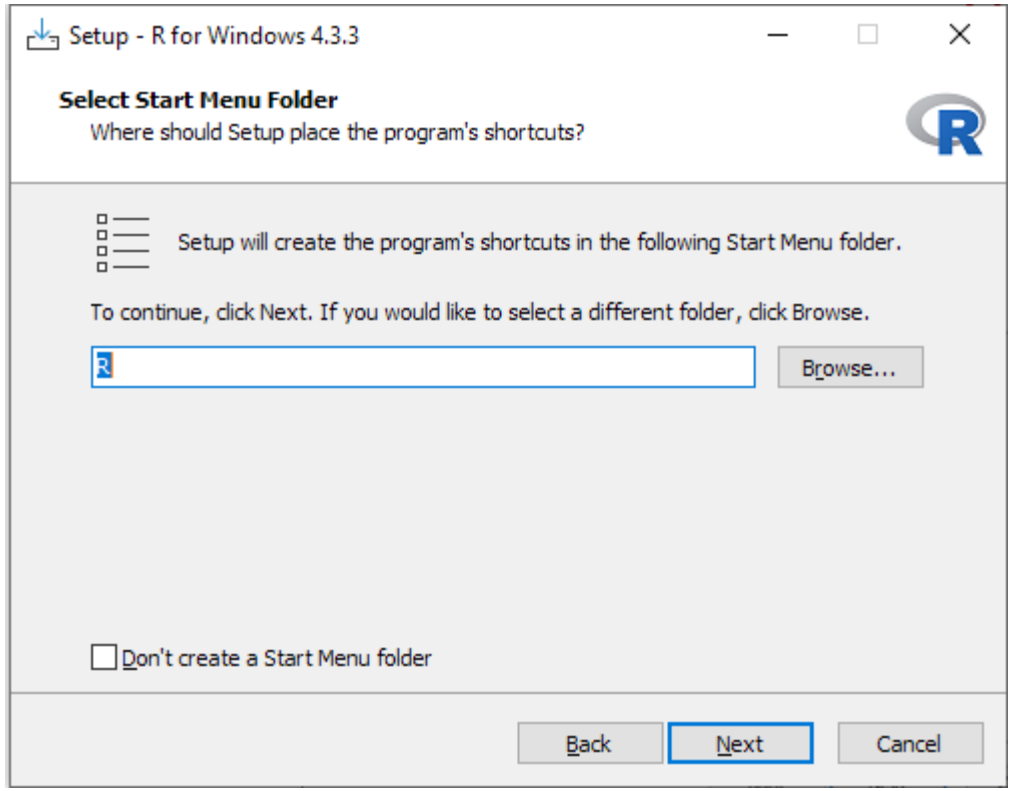
- Click Next


Setup - R for Windows 4.3.3
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
User installation
Main Files 89.7 MB
64-bit Files 71.1 MB
Message translations 8.8 MB
Current selection requires at least 172.5 MB of disk space.
Back
Next
Cancel

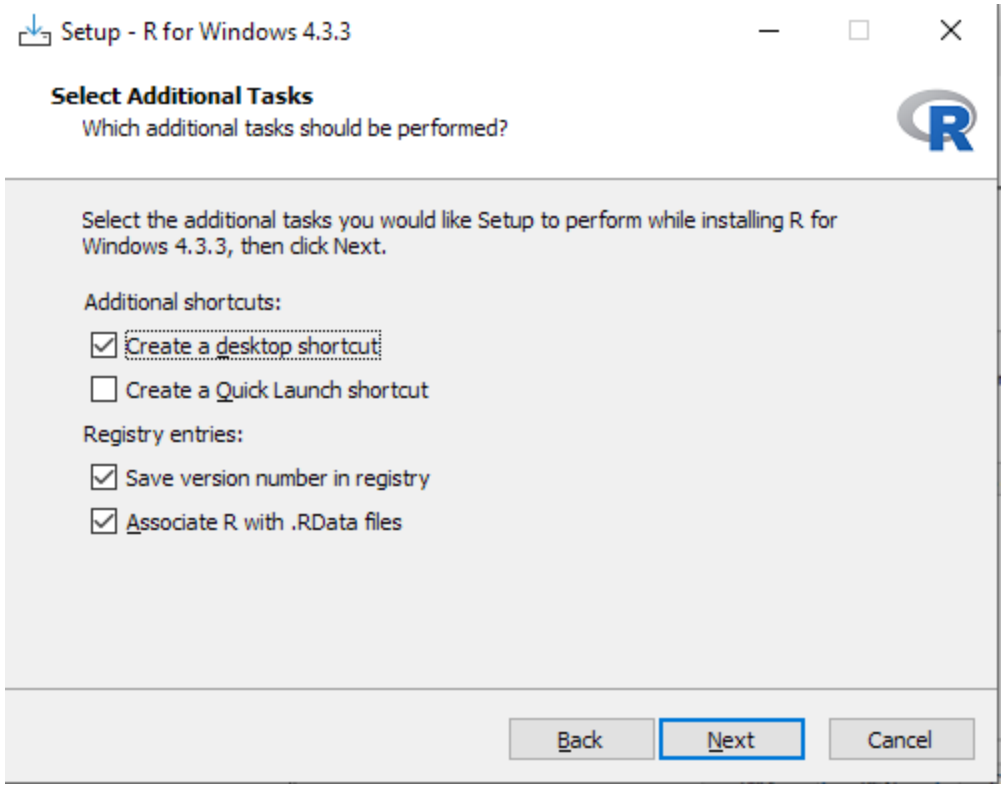
- Click Next


Setup - R for Windows 4.3.3
Startup options
Do you want to customize the startup options?
Please specify yes or no, then click Next.
Yes (customized startup)
No (accept defaults)
Back
Next
Cancel

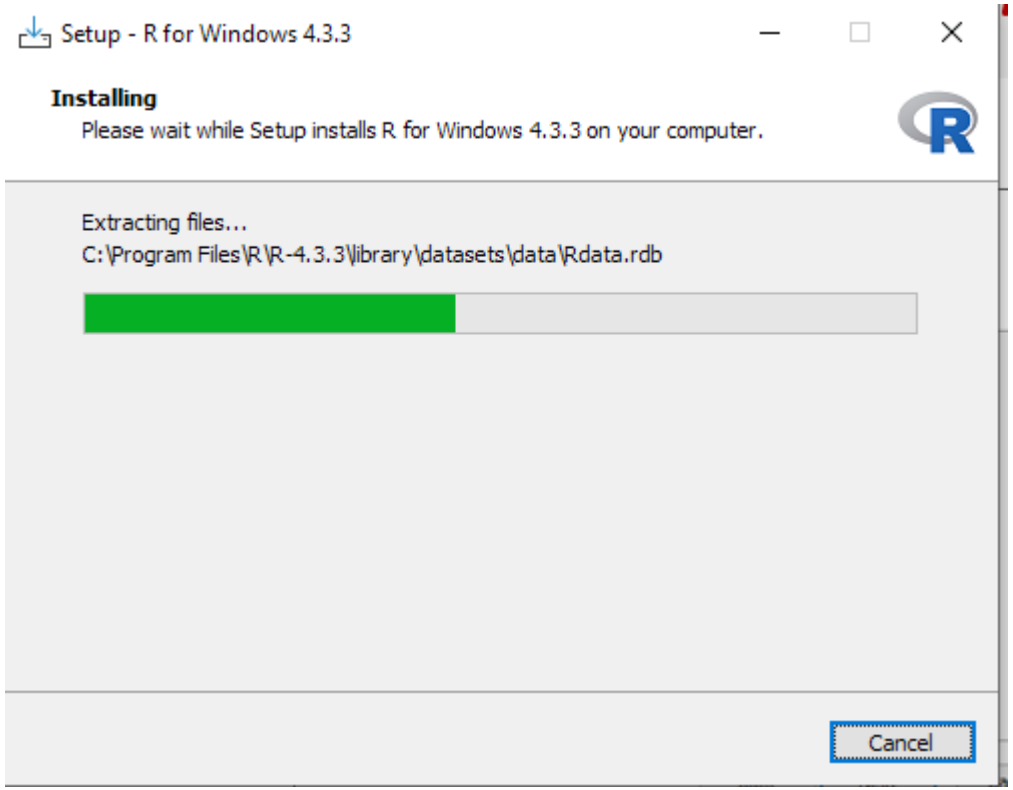
- Click Next


Setup - R for Windows 4.3.3
Select Start Menu Folder
Where should Setup place the program's shortcuts?
Setup will create the program's shortcuts in the following Start Menu folder.
To continue, click Next. If you would like to select a different folder, click Browse.
Browse...
Don't create a Start Menu folder
Back
Next
Cancel

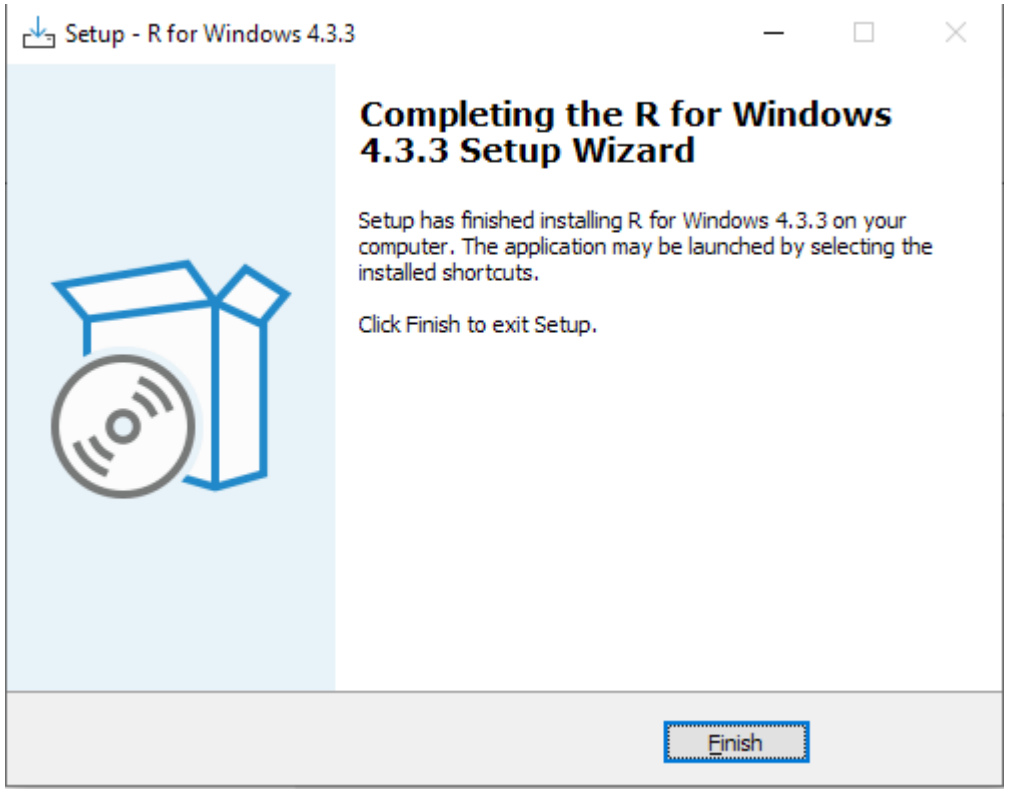
- Click Next


Setup - R for Windows 4.3.3
Select Additional Tasks
Which additional tasks should be performed?
Select the additional tasks you would like Setup to perform while installing R for Windows 4.3.3, then click Next.
Additional shortcuts:
Create a desktop shortcut
Create a Quick Launch shortcut
Registry entries:
Save version number in registry
Associate R with .RData files
Back
Next
Cancel

- It is installing R-4.3.3



- Click Finish



R INSTALLATION ON UBUNTU

- Run the following commands in Ubuntu Terminal

```
# update indices
sudo apt update -qq
# install two helper packages we need
sudo apt install --no-install-recommends software-properties-common dirmngr
# add the signing key (by Michael Rutter) for these repos # To verify key, run gpg --show-keys
/etc/apt/trusted.gpg.d/cran_ubuntu_key.asc # Fingerprint: E298A3A825C0D65DFD57CBB651716619E084DAB9
wget -qO- https://cloud.r-project.org/bin/linux/ubuntu/marutter_pubkey.asc | sudo tee -a
/etc/apt/trusted.gpg.d/cran_ubuntu_key.asc
# add the R 4.0 repo from CRAN -- adjust 'focal' to 'groovy' or 'bionic' as needed
sudo add-apt-repository "deb https://cloud.r-project.org/bin/linux/ubuntu $(lsb_release -cs)-
cran40/"
sudo apt install --no-install-recommends r-base
```

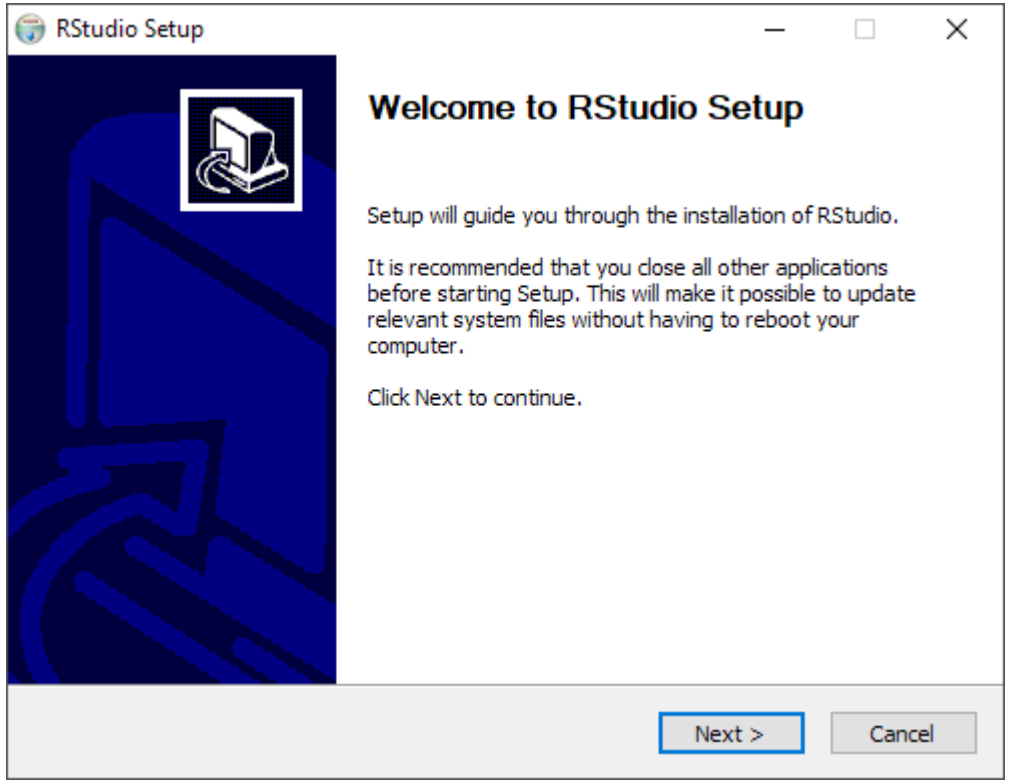
R STUDIO

Downloading R Studio

- Visit the following website
- RStudio 2021.09.0 Update: What's New – Posit
- https://posit.co/blog/rstudio-2021.09.0-update-whats-new/
- For Windows, download the following binaries or latest: https://download1.rstudio.org/electron/windows/RStudio-2023.12.1-402.exe

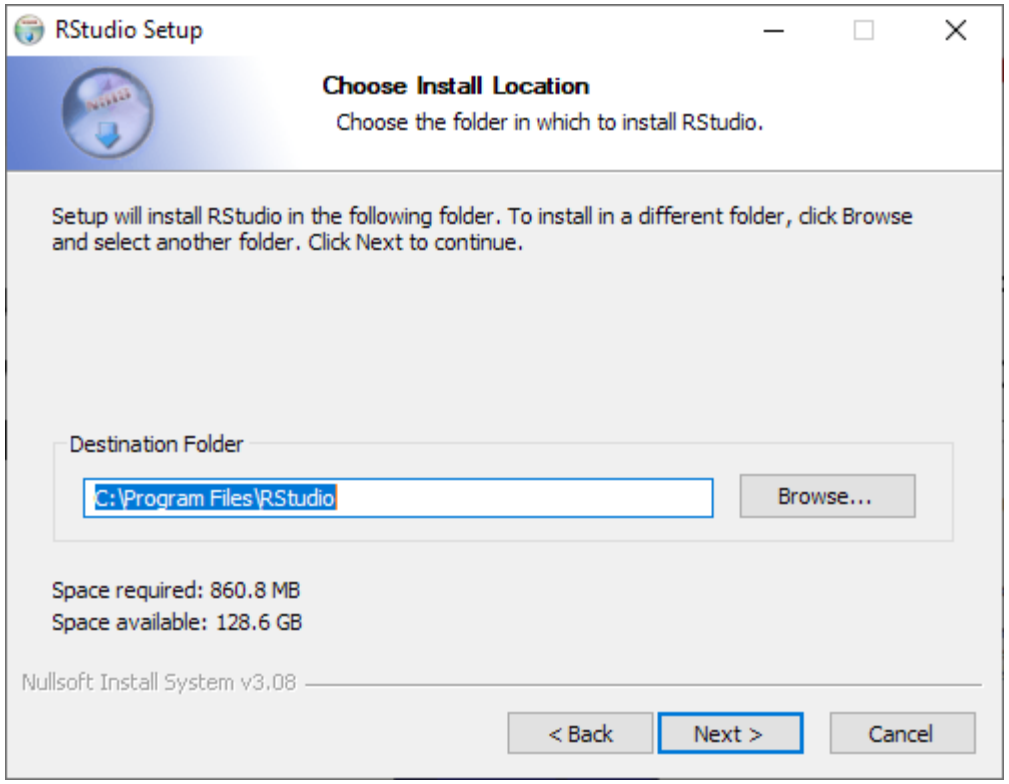
Installation of R Studio on Windows

- Double click the RStudio-2023.12.1-402.exe file from Downloads directory (or where it was downloaded)
- Click Next



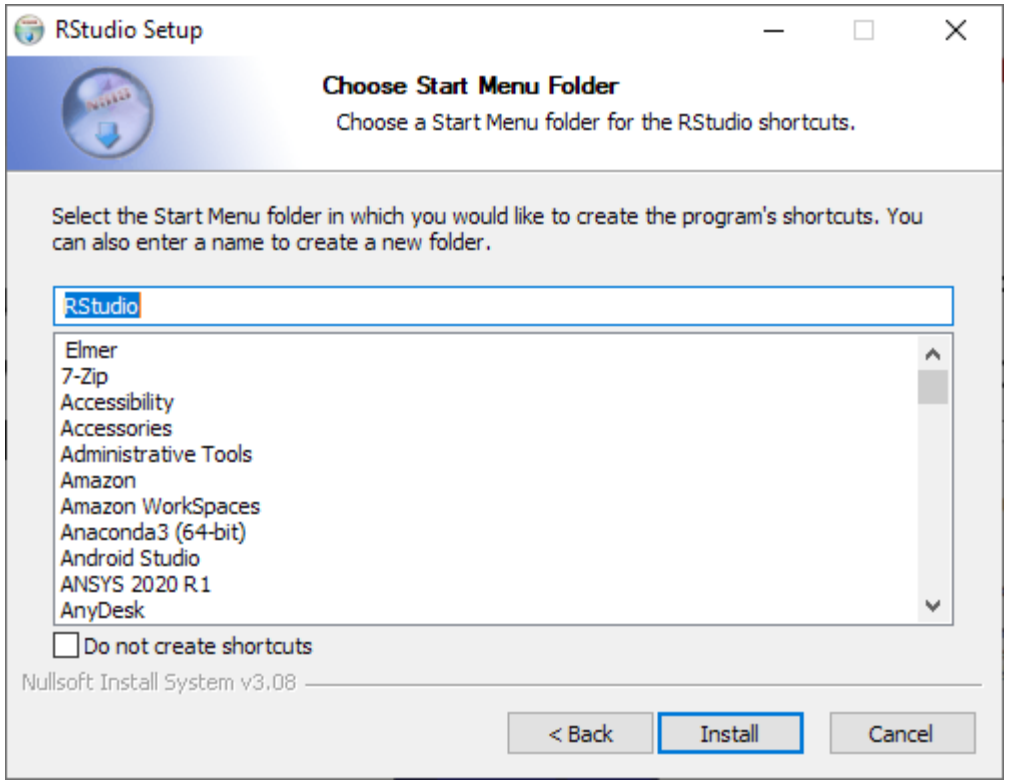
Installation of R Studio on Windows

- Click Next

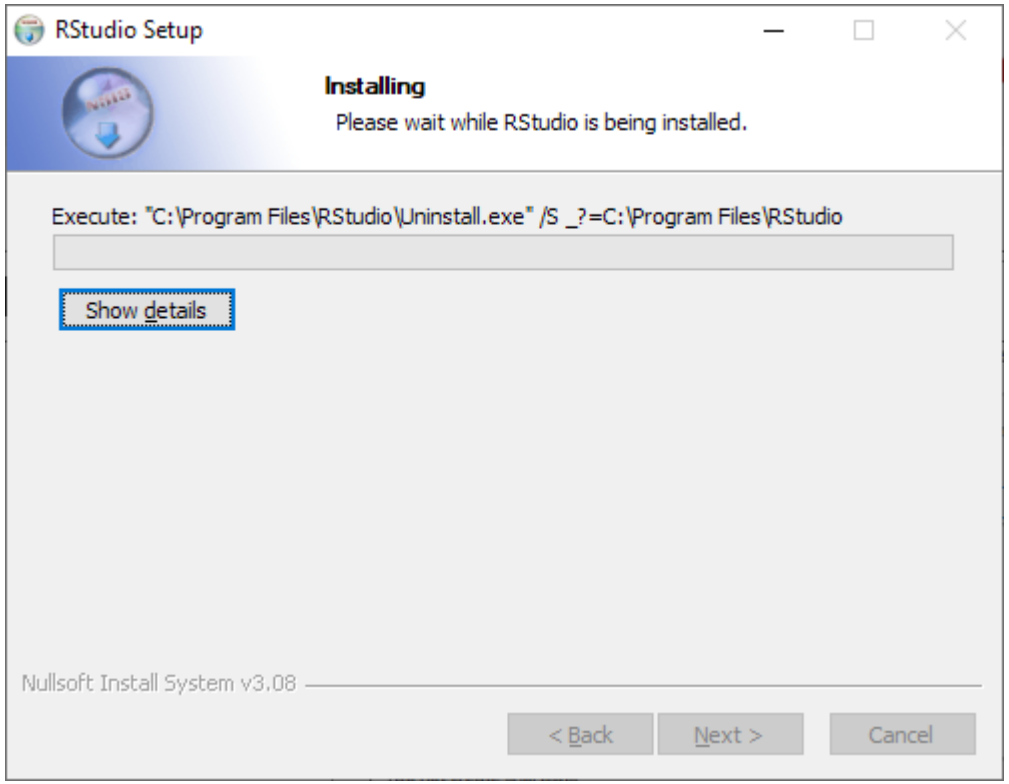


Installation of R Studio on Windows

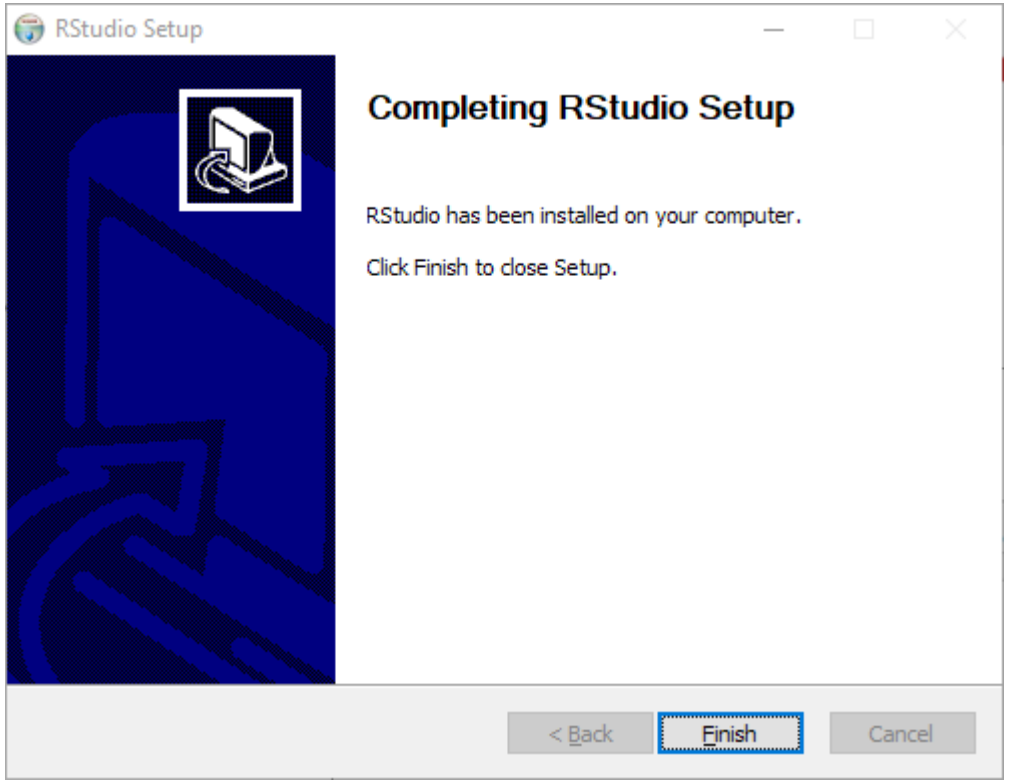
- Click Install



- Installing is going on

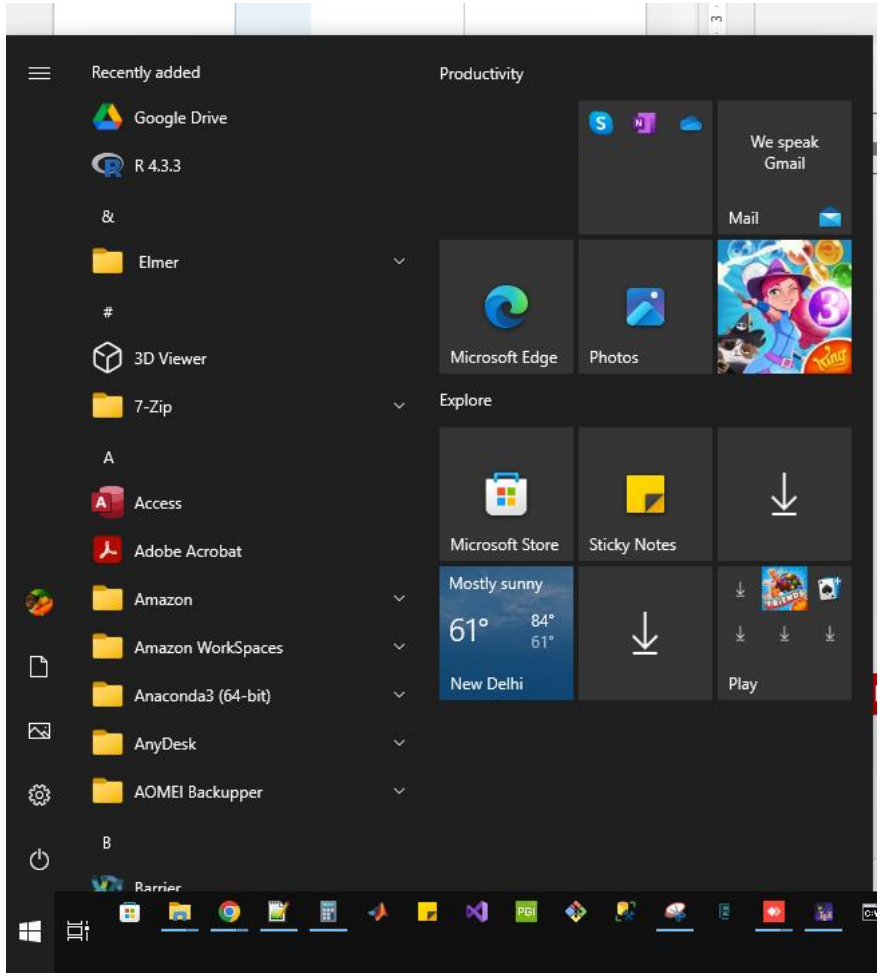


- Click Finish. Installation Completed

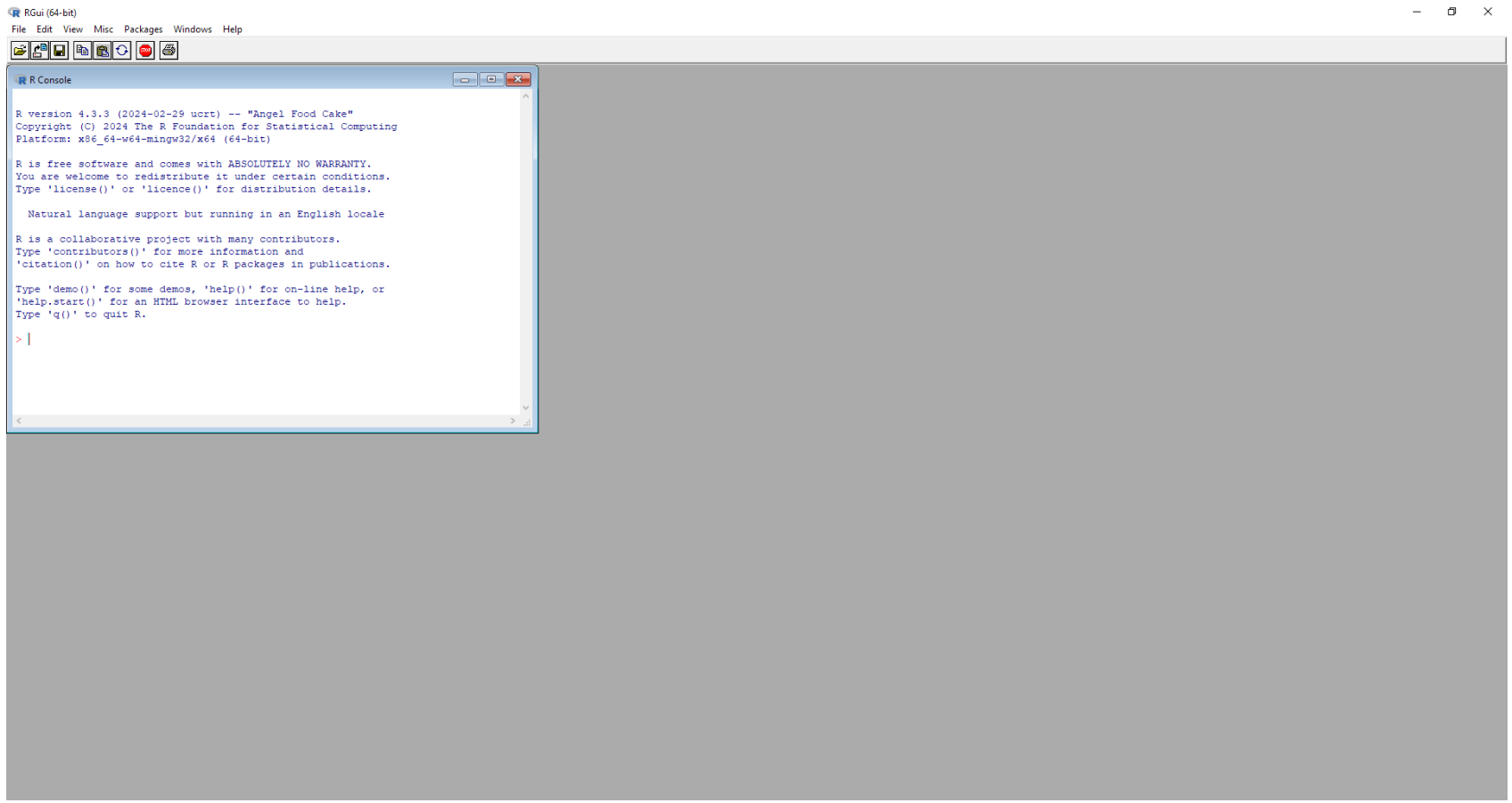


R ENVIRONMENT

- Click R 4.3.3 or latest

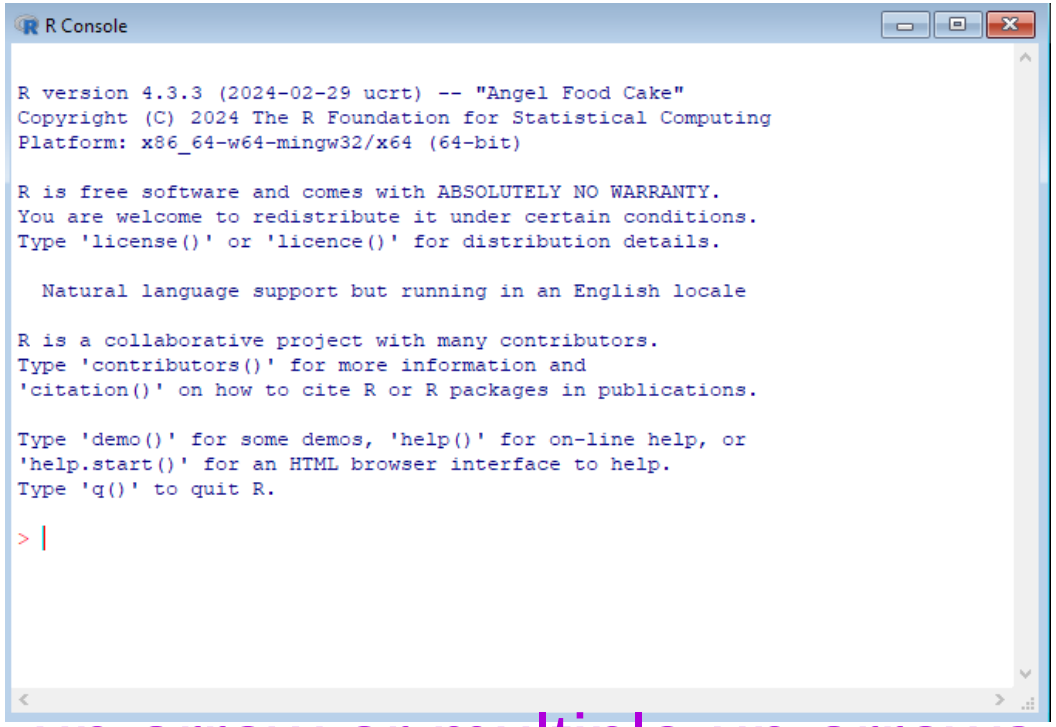


- You get the R console like this



- R is very interactive
- Results can be seen on command at a time
- The state of the objects and results can be seen at any point in R

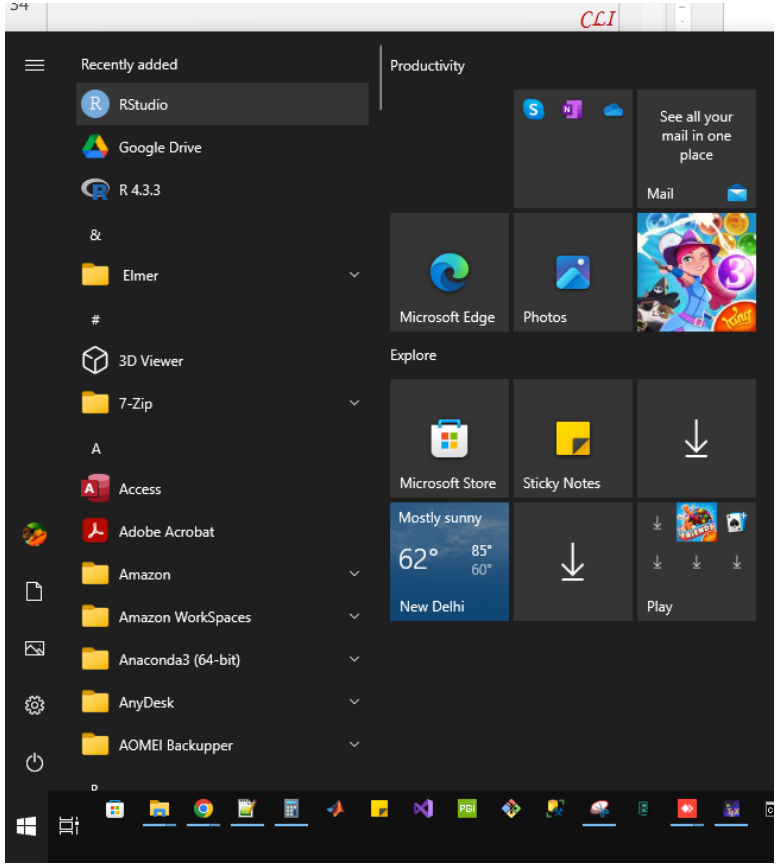
- Command Line Interface (CLI) make R so powerful
- However, frustrating to learn
- To run a command in R, type it into the console next to the > symbol and press enter



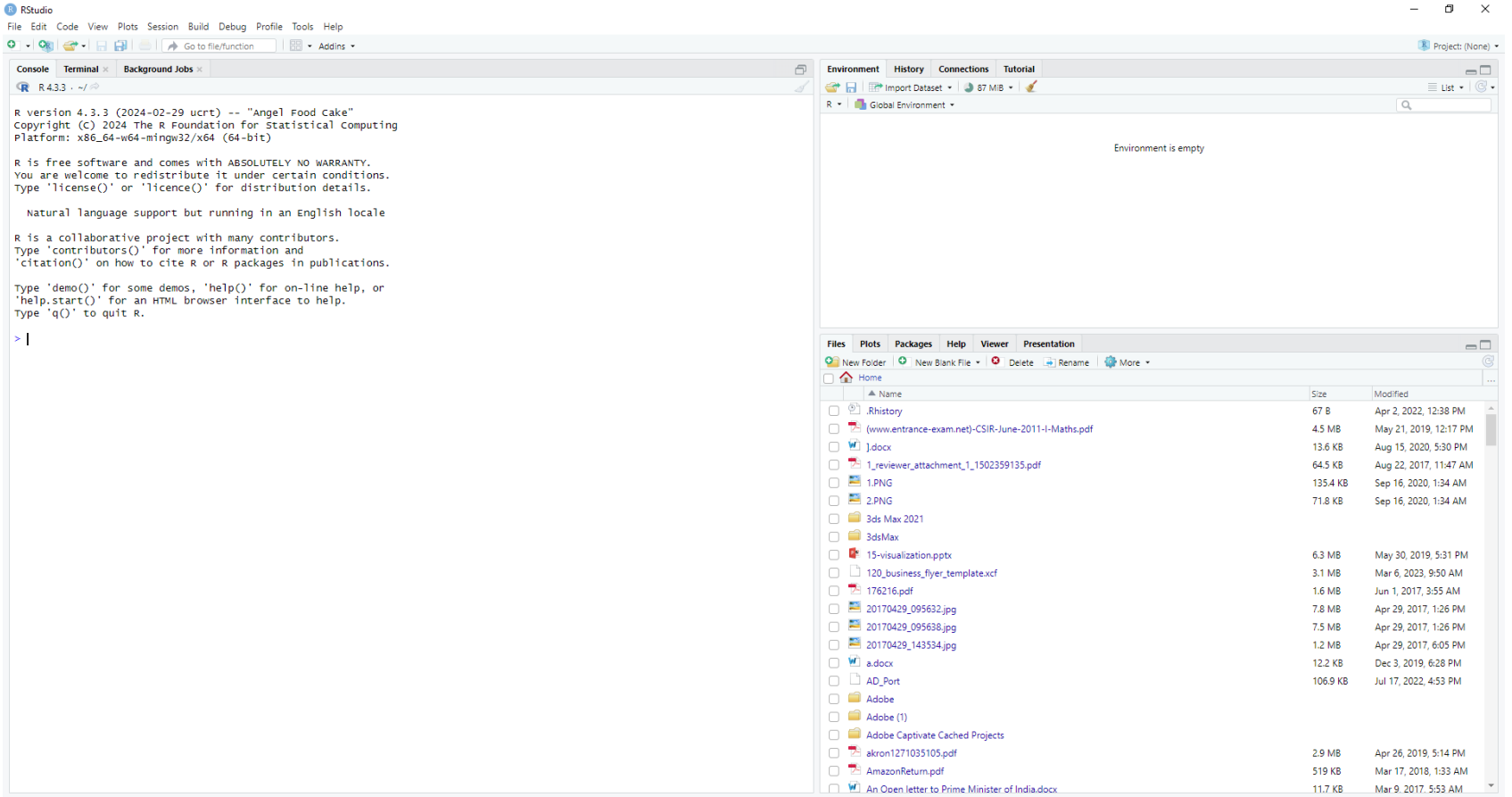
- To repeat press up arrow or multiple up arrows and hit Enter again

R STUDIO ENVIRONMENT

- Click Rstudio from the windows menu

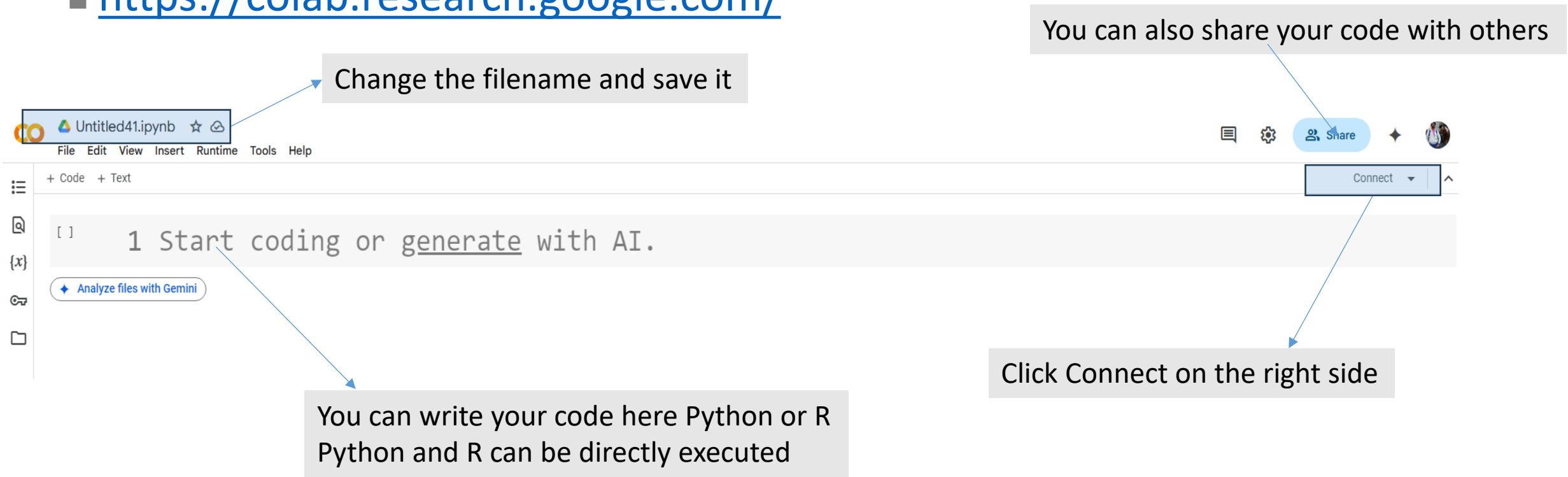


- The basic interface of Rstudio looks like the following



R ON GOOGLE COLAB

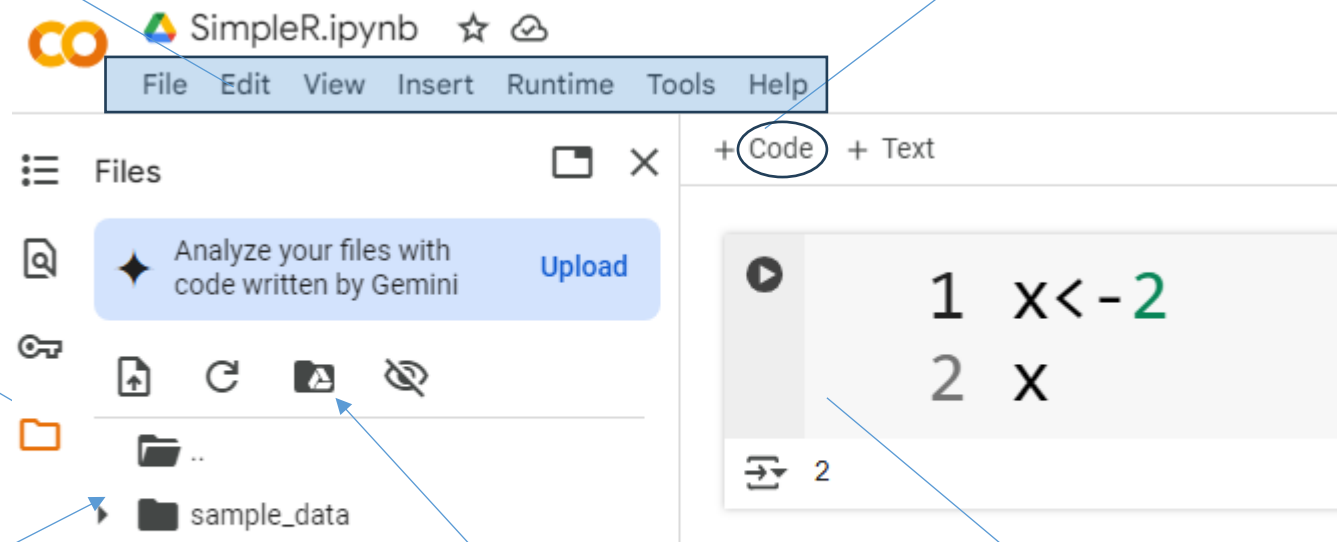
- Use your institute login and open Google Colab
- https://colab.research.google.com/



Menus

Click here for new code

Your data is saved in this folder.

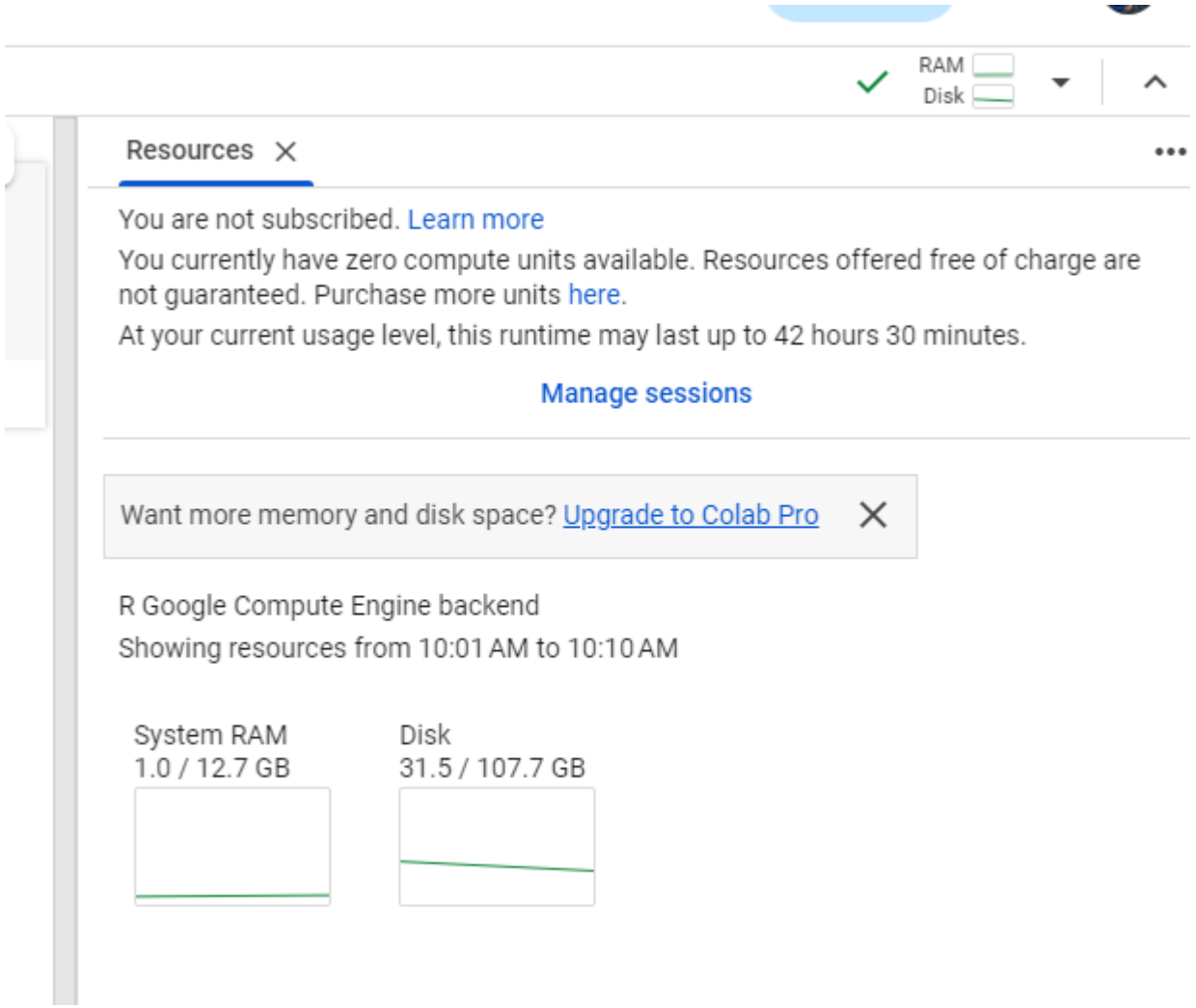


Any output saved in this folder as a file is lost after you close the browser or exit from google Colab

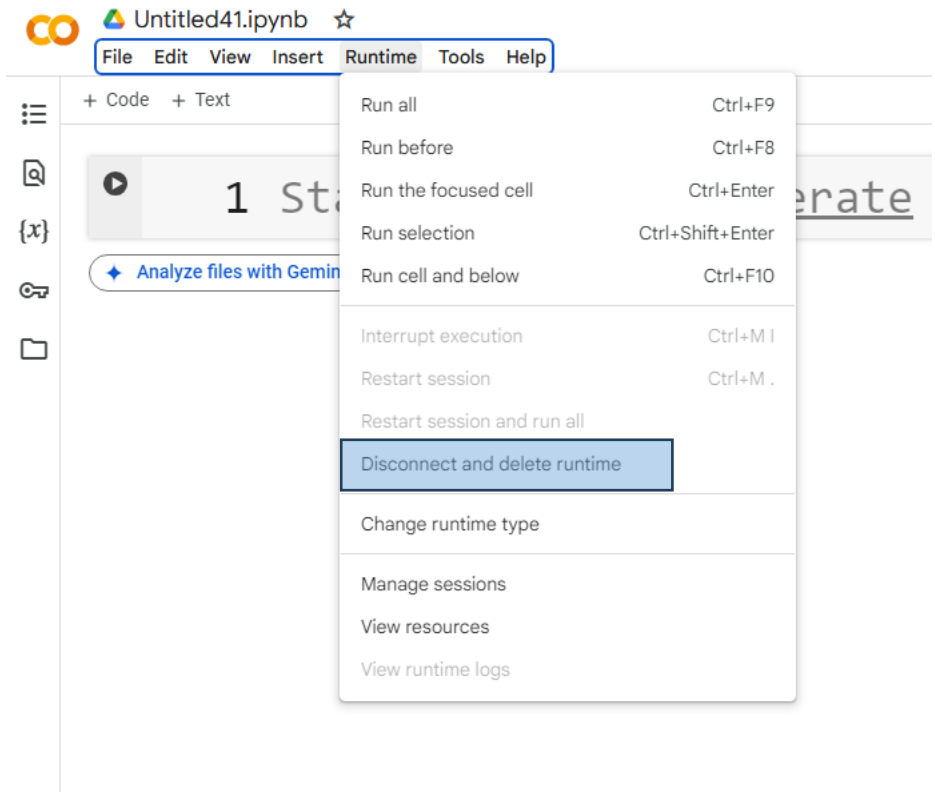
Code is save by default

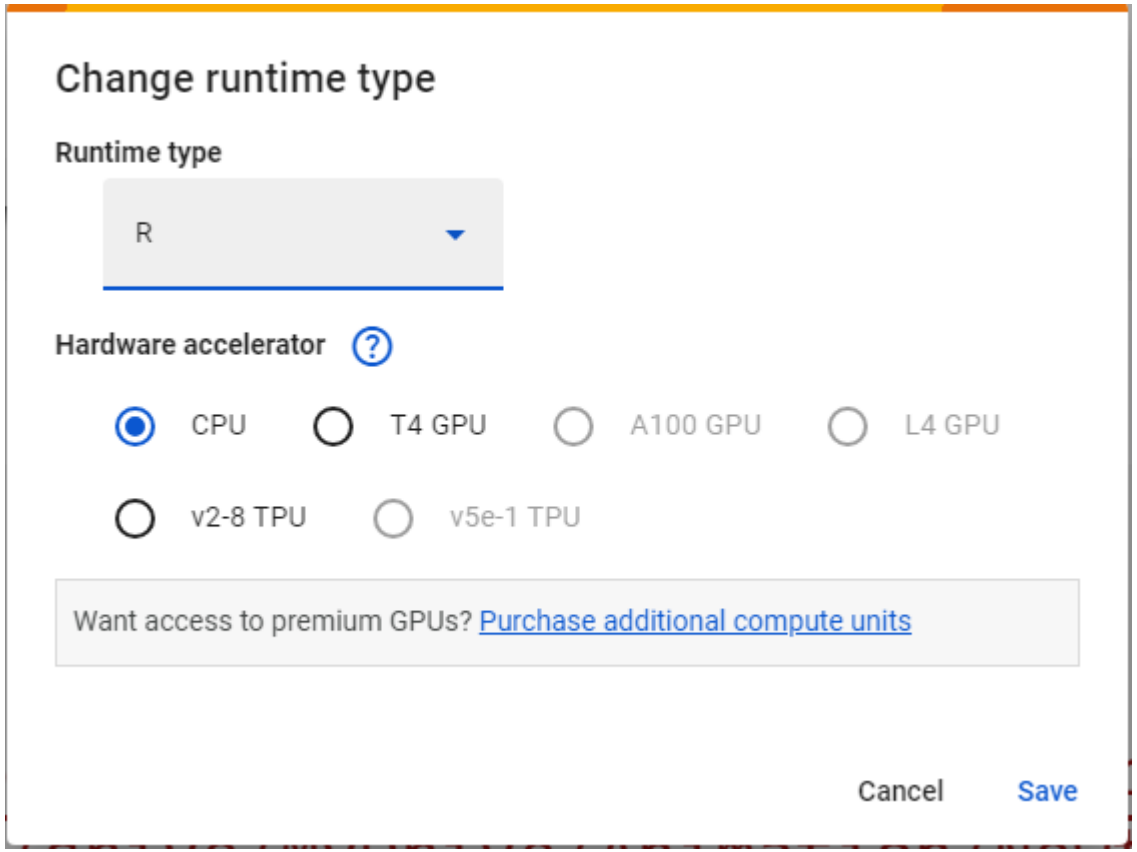
Use Google drive to store in your GDrive

By default, you get 13 GB RAM and 108 GB



- To use R,
- Select Runtime menu and then select Change Runtime Type
- Change runtime type as R





BASICS OF R

Simple Calculation

```
> 1+4
[1] 5
> 4*5
[1] 20
> 6/7
[1] 0.8571429
> 8%3
Error: unexpected input in "8%3"
> 8%%3
[1] 2
>
```

Assignment Operators

```
> x<-2
> x
[1] 2
> y=8
> y
[1] 8
> 5->x
> x
[1] 5
```

The arrow operator <- can assign a value to the variable
= is also an assignment operator

- Equal to the operator (=)
- Leftwards Operator (<-)
- Rightwards Operator (->)

Assignment Operators

```
> x<-<z-2
> x
[1] 2
> z
[1] 2
>
```

- The arrow operator <- can also assign a value to multiple variables
- Most common form of assignment is the left arrow <-

- ✓ Numbers and letters
- ✓ Dot and underscore are allowed in variable name
- × You can not start the name with a number or underscore
- × Keywords are also not allowed

Removing Variables

- You can also remove variables in R

```
> x<-2
> x
[1] 2
> y=8
> y
[1] 8
> rm(x)
> x
Error: object 'x' not found
>
```

DATA TYPES

- Numeric
- Character (String)
- Date/POSIXCT
- Logical

```
> i<-5L
> class(i)
[1] "integer"
```

```
> x<-5.2
> class(x)
[1] "numeric"
```

```
> x<-5L/2L
> class(x)
[1] "numeric"
```

```
> i=5L
> is.numeric(i)
[1] TRUE
> is.integer(i)
[1] TRUE
```

```
> x=6L/4L
> is.numeric(x)
[1] TRUE
> is.integer(x)
[1] FALSE
> x
[1] 1.5
```

```
> x=9+3i
> class(x)
[1] "complex"
```

```
> x<-"Data Science"
> class(x)
[1] "character"
```

```
> x<-"Data Science"
> x
[1] "Data Science"
> y<-factor("Data Science")
> y
[1] Data Science
Levels: Data Science
```

```
> x<-"Data Science"
> nchar(x)
[1] 12
> nchar(4598)
[1] 4
> nchar(9)
[1] 1
```

Multiline strings

```
> x<-"Data Science Programming Laboratory
+ is my course"
> x
[1] "Data Science Programming Laboratory \nis my course"


> grepl("Data",x)
[1] TRUE

> x<-"Data"
> y<-"Science"
> x+y
Error in x + y : non-numeric argument to binary operator
> paste(x,y)
[1] "Data Science"
```

Escape Characters

Code	Result
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace

```
> date1<-as.Date("2024-03-12")
> date1
[1] "2024-03-12"
> class(date1)
[1] "Date"
> as.numeric(date1)
[1] 19794
```

What does as.numeric(date1) returns?

What does as.numeric(date2) returns?

```
> date2<-as.POSIXct("2024-03-13 14:23")
> date2
[1] "2024-03-13 14:23:00 IST"
> class(date2)
[1] "POSIXct" "POSIXt"
> as.numeric(date2)
[1] 1710319980
```

```
> x=TRUE
> class(x)
[1] "logical"
> y=FALSE
> class(y)
[1] "logical"
```

```
> x=TRUE
> is.logical(x)
[1] TRUE
```

```
> 4==5
[1] FALSE
> 4!=7
[1] TRUE
> 3>9
[1] FALSE
> 4<10
[1] TRUE
> 4<=11
[1] TRUE
> 9>=12
[1] FALSE
> "data"=="science"
[1] FALSE
> "data"<"science"
[1] TRUE
```

VECTORS

```
> x=c(1,2,3,4,4,5,5)
> class(x)
[1] "numeric"
> x
[1] 1 2 3 4 4 5 5
> x*3
[1]  3  6  9 12 12 15 15
> x+2
[1] 3 4 5 6 6 7 7
> x-3
[1] -2 -1  0  1  1  2  2
> x/4
[1] 0.25 0.50 0.75 1.00 1.00 1.25 1.25
> sqrt(x)
[1] 1.000000 1.414214 1.732051 2.000000 2.000000 2.236068 2.236068
>
```

Vectors

- Collection of elements, all of the same type
- Vector cannot be of mixed datatype
- R is a vectorized language
- Operations are applied to each element of the vector automatically
- Vectors do not have a dimension: Nothing called column vector or row vector
- Most common way to create a vector is with “c”
- “c” stands for combine because multiple elements are being combined into a vector

```
> y=c("Data","Science","Programming","Laboratory")
> y
[1] "Data"        "Science"     "Programming" "Laboratory"
> nchar(y)
[1]  4  7 11 10
```

- To create a sequence of numbers, we can use : operator

```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> -10:10
 [1] -10  -9  -8  -7  -6  -5  -4  -3  -2  -1   0   1   2   3   4   5   6   7
8
[20]   9  10
> 5:-8
 [1]  5  4  3  2  1  0 -1 -2 -3 -4 -5 -6 -7 -8
```

- To create a sequence of numbers, we can use : operator

```
> x<-1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> y<--5:10
> y
 [1] -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7  8  9 10
> x-y
 [1]  6  6  6  6  6  6  6  6  6  6 -4 -4 -4 -4 -4 -4
Warning message:
In x - y : longer object length is not a multiple of shorter object length
> length(x)
[1] 10
> length(y)
[1] 16
```

- To create a sequence of numbers, we can use : operator

```
> x=1:10
> y=11:20
> length(y)
[1] 10
> length(x)
[1] 10
> x+y
 [1] 12 14 16 18 20 22 24 26 28 30
> x^y
 [1] 1.000000e+00 4.096000e+03 1.594323e+06 2.684355e+08 3.051758e+10
 [6] 2.821110e+12 2.326305e+14 1.801440e+16 1.350852e+18 1.000000e+20
> x/y
 [1] 0.09090909 0.16666667 0.23076923 0.28571429 0.33333333 0.37500000
 [7] 0.41176471 0.44444444 0.47368421 0.50000000
```

- Observe the difference

```
> 2+3:6
[1] 5 6 7 8
> (2+3):6
[1] 5 6
> 1:3^2
[1] 1 2 3 4 5 6 7 8 9
> (1:3)^2
[1] 1 4 9
```

- Compare vectors

```
> x=1:10
> x<=20
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
> y=2:11
> x>y
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> x<y
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
> x=1:10
> y=-3:6
> any(x<y)
[1] FALSE
> all(x<y)
[1] FALSE
> all(x>y)
[1] TRUE
```

Vectors: Indexing and Naming

- R indexing starts with 1, you can use colon operator and c for fetching elements

```
> x=1:10
> x[1]
[1] 1
> x[1:2]
[1] 1 2
> x[c(1,4)]
[1] 1 4
```

- You can provide name for each element of the vector in two ways

```
> x=c(MA522M="Data",MA523L="Stochastic",MA501L="Multi")
> x
     MA522M       MA523L       MA501L
     "Data" "Stochastic"      "Multi"
> x=1:5
> names(x)=c("one","two","three","four","five")
> x
  one   two three  four  five
    1     2     3     4     5
```

Factor Vectors

- Another important concept in R
- Used to handle **categorical variables**
- Store them efficiently
- Easy statistical analysis and plotting
- Find the unique value of the variable: Levels

```
> x=c("Raja","Ravi","Rani","Ramya","Raghu","Raghu","Ramya")
> x
[1] "Raja"  "Ravi"  "Rani"  "Ramya" "Raghu" "Raghu" "Ramya"
> y=as.factor(x)
> y
[1] Raja  Ravi  Rani  Ramya Raghu Raghu Ramya
Levels: Raghu Raja Ramya Rani Ravi
```

```
> as.numeric(y)
[1] 2 5 4 3 1 1 3
```

- Technically R is giving each unique value of a factor a unique integer tying it back to the character representation

- It reduces the size of the variable, because they are sorting only the unique values
- However, it can create a headache if not handled properly

```
> factor(x=c("Nursery","Elementary School","Middle School","High
School","College"),
+ levels=c("Nursery","Elementary School","Middle School","High
School","College"),ordered=TRUE)
[1] Nursery           Elementary School Middle School     High School
[5] College
5 Levels: Nursery < Elementary School < Middle School < ... < College
```

CALLING FUNCTIONS

- Any built-in function provided in R has accompanying documentation

```
> x<-c(1,2,4,5,6)
> mean(x)
[1] 3.6
```

```
> ? mean
starting httpd help server ... done
```

```
> apropos("mea")
 [1] ".colMeans"          ".rowMeans"          "colMeans"
 [4] "influence.measures" "kmeans"             "mean"
 [7] "mean.Date"          "mean.default"       "mean.difftime"
[10] "mean.POSIXct"       "mean.POSIXlt"       "rowMeans"
[13] "weighted.mean"
```

MISSING DATA

NA and NULL

- Missing data plays a crucial role in both statistics and computing
- NA and NULL are two types of missing data
- When data missing such as a dash, a period or the number, R uses NA
- NA is often used as another element of a vector

```
> x<-c(1,2,NA,8,3,NA,3)
> x
[1]  1  2 NA  8  3 NA  3
> is.na(x)
[1] FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
```

NA and NULL

- NULL is the absence of anything
- It is not exactly missingness, but nothingness
- NULL is atomical and cannot exist within a vector
- If used inside a vector it simply disappears

```
> x<-c(1,NULL,4)
> x
[1] 1 4
> a=NULL
> is.null(a)
[1] TRUE
> is.null(x)
[1] FALSE
```

OPERATORS

Arithmetic Operators

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
^	Exponent	x ^ y
%%	Modulus (Remainder from division)	x %% y
%/%	Integer Division	x%/%y

```
> x=4
> y=3
> x+y
[1] 7
> x-y
[1] 1
> x*y
[1] 12
> x/y
[1] 1.333333
> x^y
[1] 64
> x%%y
[1] 1
> x%/%y
[1] 1
```

Assignment Operators

<<- is a global assigner

```
> x<-3
> x
[1] 3
> x<<-3
> x<<-4
> x
[1] 4
> 3->x
> 3->>x
```

Comparison Operators

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical Operators

Operator	Description
&	Element-wise Logical AND operator
&&	Logical AND operator
|	Elementwise- Logical OR operator
||	Logical OR operator
!	Logical NOT

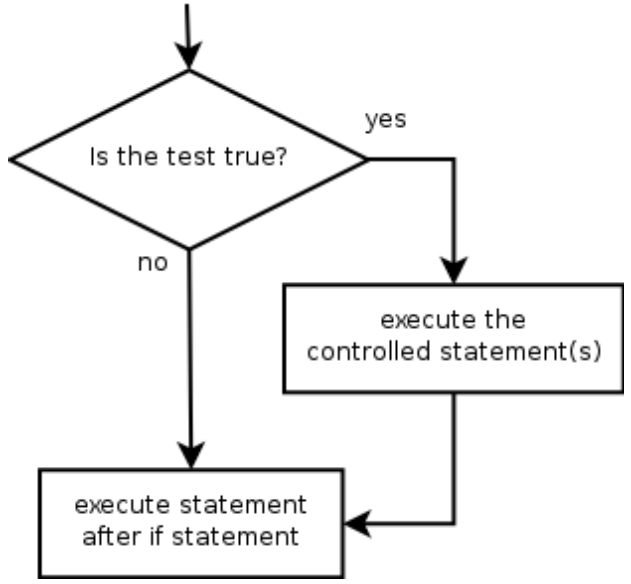
Other Operators

Operator	Description
:	Series of Numbers
%in%	Check elements in a vector
%*%	Matrix Multiplication

BRANCHING

- **`if` statement**: Executes a group of statements only if a certain condition is true. Otherwise, the statements are skipped.

  - Syntax:
    ```
    if (condition) {
        statements
    }
    ```

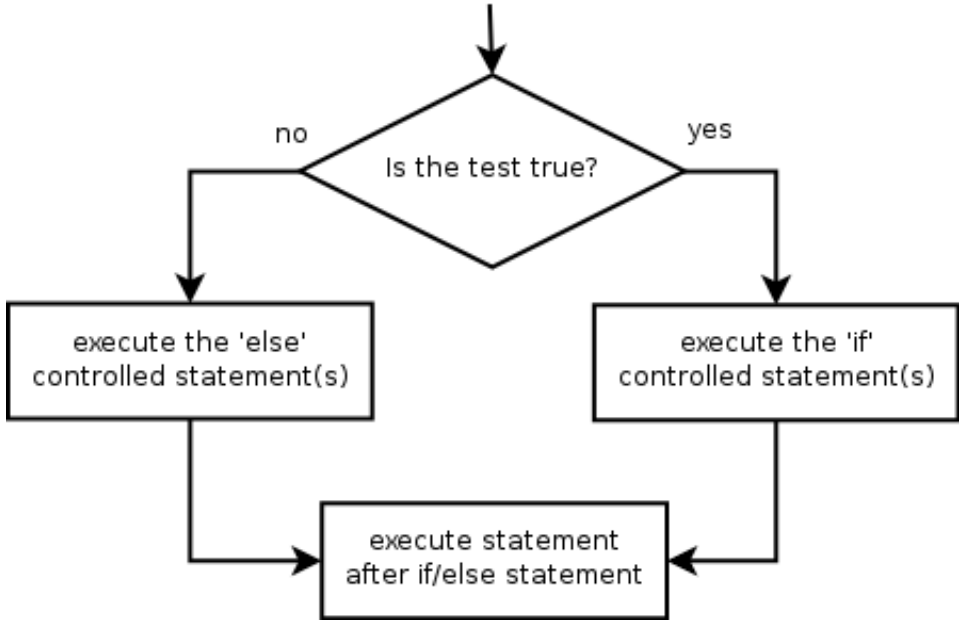


- Example:

```
gpa <- 3.4
if gpa > 2.0{
    print("Your application is accepted.")
}
```

- **`if/else` statement**: Executes one block of statements if a certain condition is True, and a second block of statements if it is False.
  - Syntax:

```
if (condition) {
        statements
}
else{
        statements
}
```



- Example:

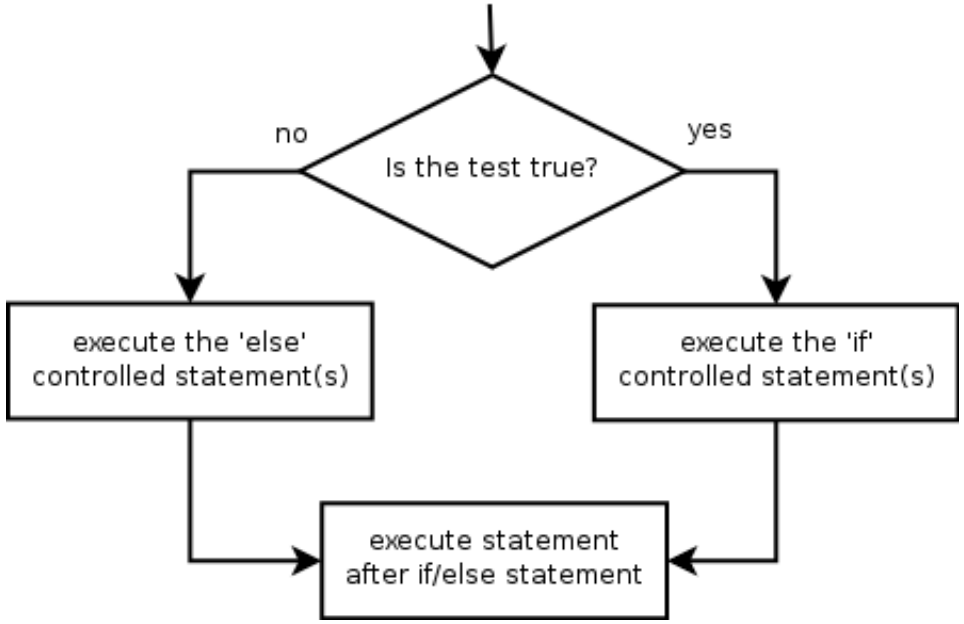
```
gpa = 8.4
if (gpa > 7.0){
    print("Welcome to Kathmandu University!")
}
else{
    print("Your application is denied.")
 }
```

- **`if/else` statement**: Executes one block of statements if a certain condition is True, and a second block of statements if it is False.
  - Syntax:

```
if (condition) {
        statements
}
else{
        statements
}
```

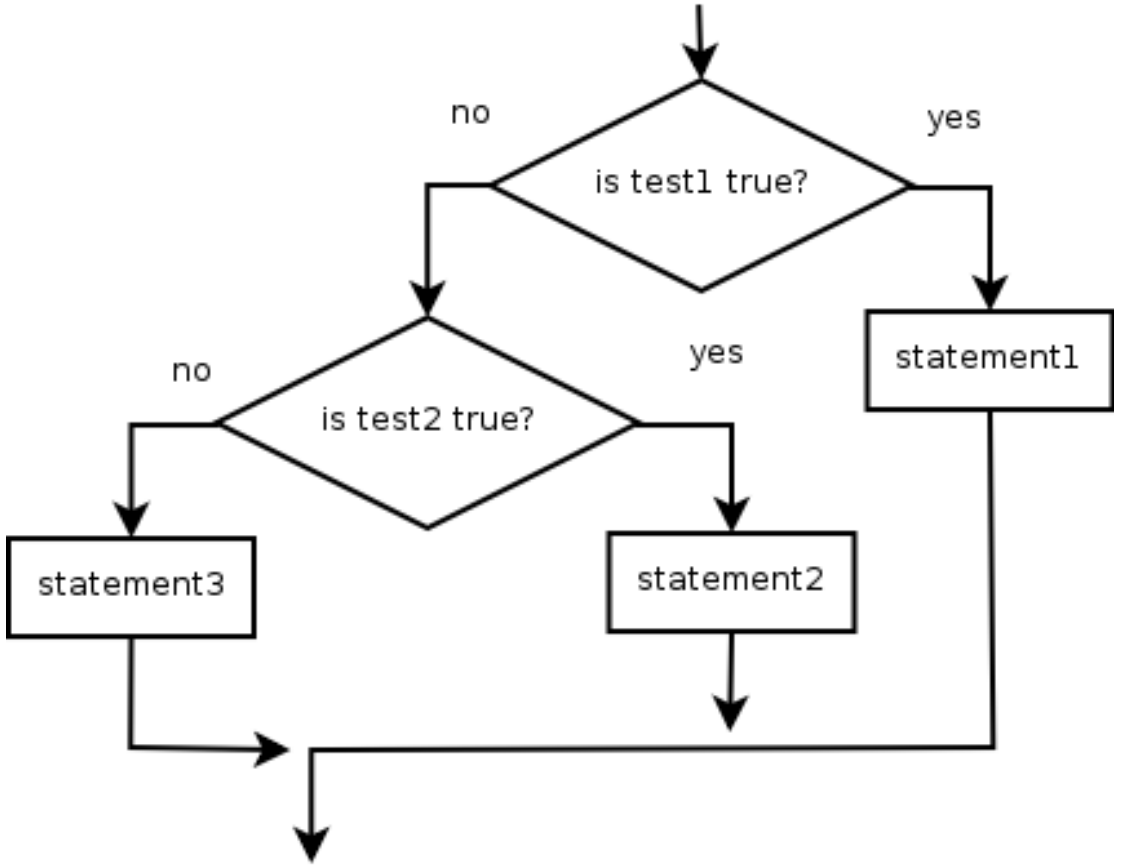
- Example:

```
check = 4.0/2.0
if (check==round(check)){
        print("Integer")
}
else{
        print("Not Integer.")
 }
```



- Multiple conditions can be chained with "else if":

```
if (condition) {
    statements}
else if (condition) {
    statements}
else {
    statements}
```



Comparison Operators

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical Operators

Operator	Description
&	Element-wise Logical AND operator
&&	Logical AND operator
|	Elementwise- Logical OR operator
||	Logical OR operator
!	Logical NOT

LOOPS

- **`for` loop**: Repeats a set of statements over a group of values.
  - Syntax:

```
for (x in vectors){
    statements
}
```

- Example:

```
for (x in 1:5){
    print(x*x)
 }
```

```
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
```

- Example:

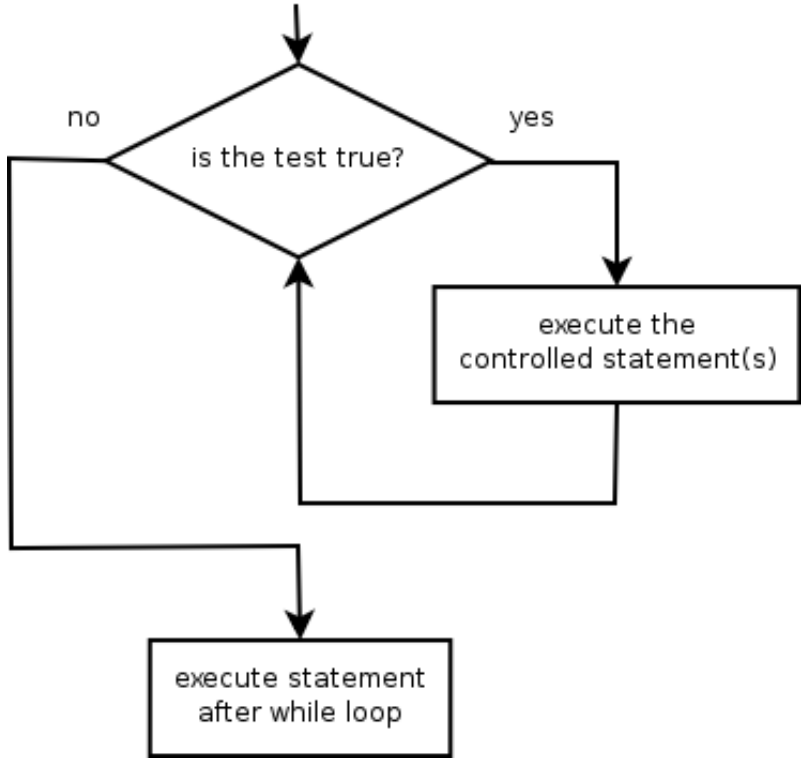
```
name=c("MA23M001","MA23M002","MA23M003","MA
23M004","MA23M005")

for (x in name){
    print(x)
 }
```

```
[1] "MA23M001"
[1] "MA23M002"
[1] "MA23M003"
[1] "MA23M004"
[1] "MA23M005"
```

- **`while` loop**: Executes a group of statements as long as a condition is True.
  - good for *indefinite loops* (repeat an unknown number of times)

- Syntax:

```
while (condition)
      statements
```

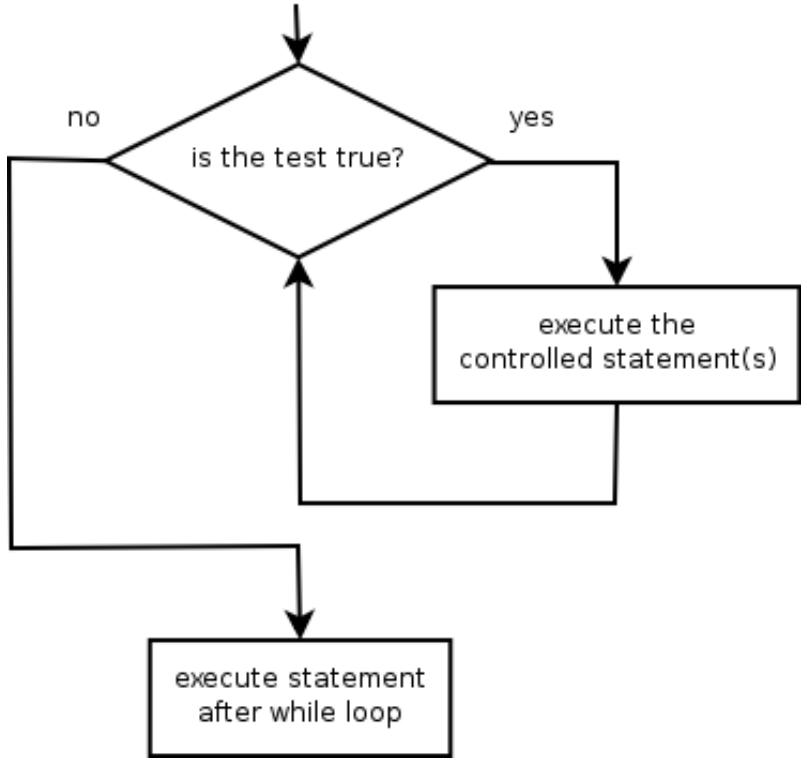


- Example:

```
number = 1
while (number < 200){
    print(number)
    number = number * 2
}
```

- Output:

```
1 2 4 8 16 32 64 128
```



End of Introduction to R