

SIRE516

Week1: Introduction to R programming

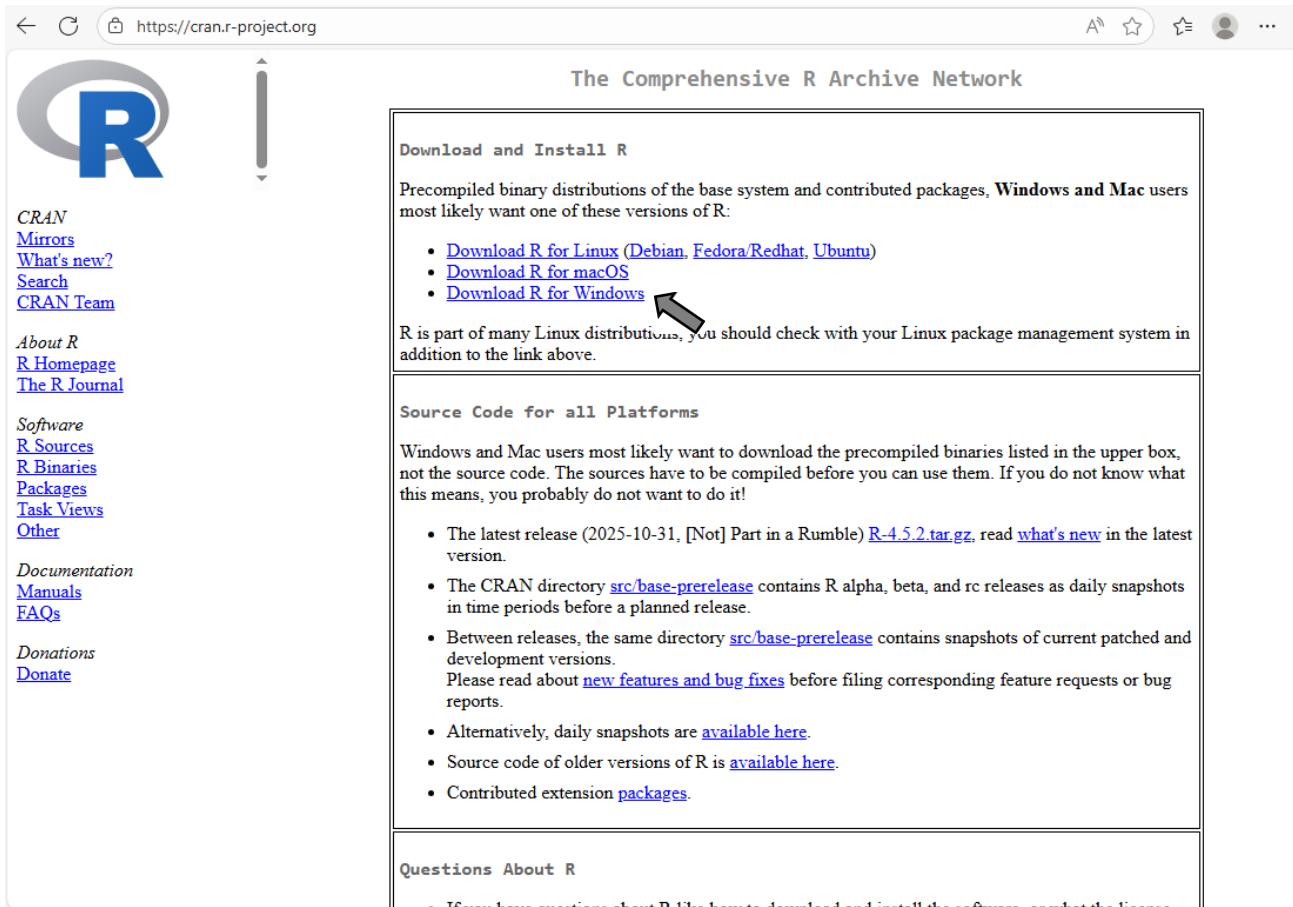
Let's get R

- Download the latest version of R at <https://cran.r-project.org/>

The screenshot shows the main page of The Comprehensive R Archive Network (CRAN). On the left, there is a sidebar with links to 'CRAN Mirrors', 'What's new?', 'Search', and 'CRAN Team'. Below that are sections for 'About R' (links to 'R Homepage' and 'The R Journal'), 'Software' (links to 'R Sources', 'R Binaries', 'Packages', 'Task Views', and 'Other'), 'Documentation' (links to 'Manuals' and 'FAQs'), and 'Donations' (link to 'Donate'). The main content area has a large title 'The Comprehensive R Archive Network'. Below it is a box titled 'Download and Install R' which contains text about precompiled binary distributions for Windows and Mac users, followed by a bulleted list of download links for Linux, macOS, and Windows. Another box titled 'Source Code for all Platforms' provides information about source code availability and compilation, with a detailed bulleted list of options for different release types and sources. A third box at the bottom is titled 'Questions About R'.

Let's get R

- Choose your OS



The screenshot shows the main page of the Comprehensive R Archive Network (CRAN) at <https://cran.r-project.org>. The page features a large R logo on the left and a navigation menu with links like 'CRAN', 'Mirrors', 'What's new?', 'Search', 'CRAN Team', 'About R', 'R Homepage', 'The R Journal', 'Software', 'R Sources', 'R Binaries', 'Packages', 'Task Views', 'Other', 'Documentation', 'Manuals', 'FAQs', 'Donations', and 'Donate'. The central content area is titled 'The Comprehensive R Archive Network' and contains three main sections: 'Download and Install R', 'Source Code for all Platforms', and 'Questions About R'. The 'Download and Install R' section is highlighted with a red arrow pointing to the 'Download R for Windows' link. This section also includes a note about Linux distributions and links for Windows, Mac, and Linux users. The 'Source Code for all Platforms' section provides instructions for Windows and Mac users regarding source code compilation and daily snapshots. The 'Questions About R' section contains a single bullet point about software licensing.

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows** and **Mac** users most likely want one of these versions of R:

- [Download R for Linux \(Debian, Fedora/Redhat, Ubuntu\)](#)
- [Download R for macOS](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2025-10-31, [Not] Part in a Rumble) [R-4.5.2.tar.gz](#), read [what's new](#) in the latest version.
- The CRAN directory [src/base-prerelease](#) contains R alpha, beta, and rc releases as daily snapshots in time periods before a planned release.
- Between releases, the same directory [src/base-prerelease](#) contains snapshots of current patched and development versions.
Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Alternatively, daily snapshots are [available here](#).
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#).

Questions About R

- If you have questions about R like how to download and install the software, or what the license

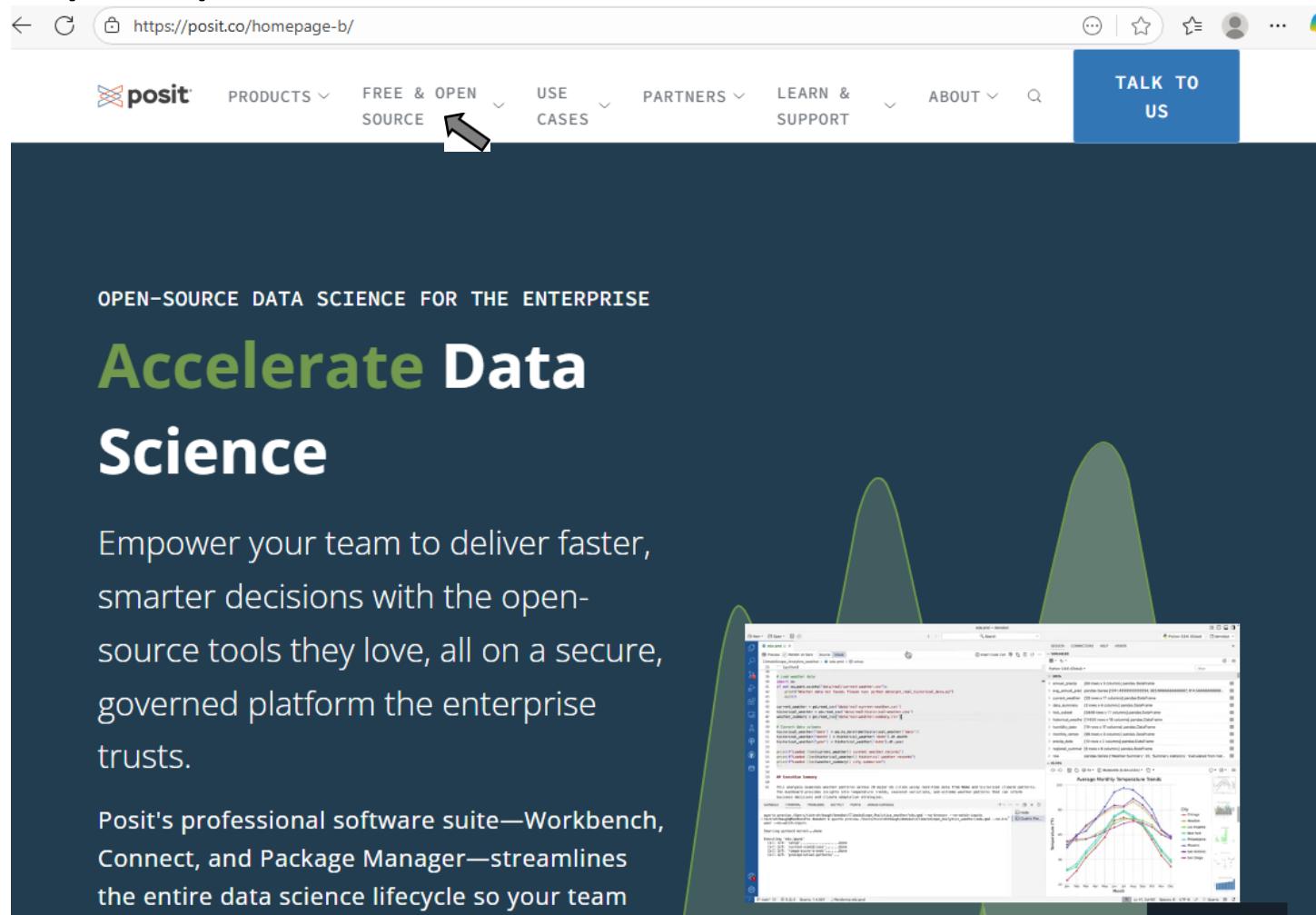
RStudio

- Integrated development environment (IDE) for R
- Organizing your R coding environment
- Facilitate datasets loading
- Facilitate graph and plot resizing and saving

*****ALWAYS install R before Rstudio*****

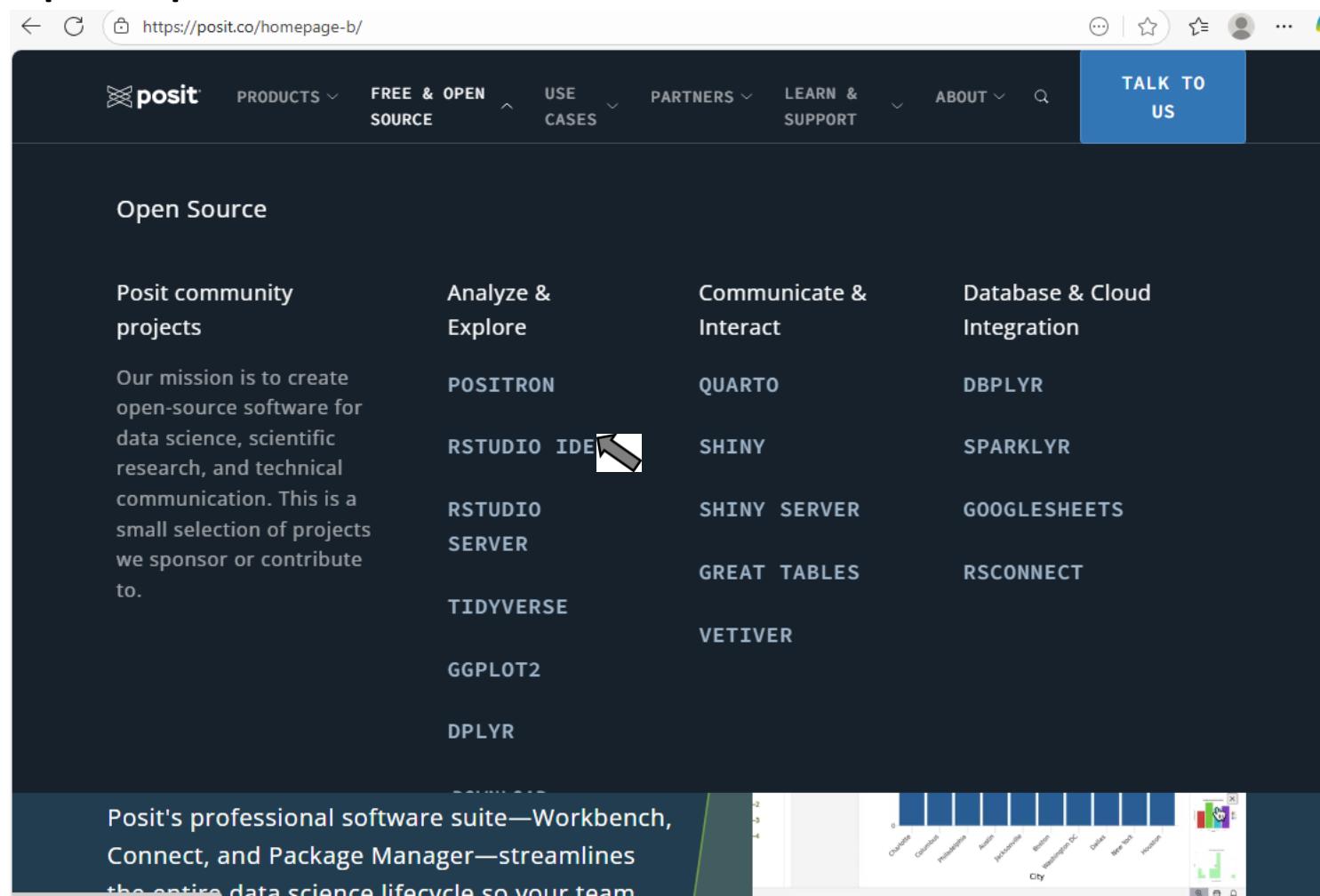
Let's get RStudio

- Download the latest version of RStudio at
<https://posit.co/>



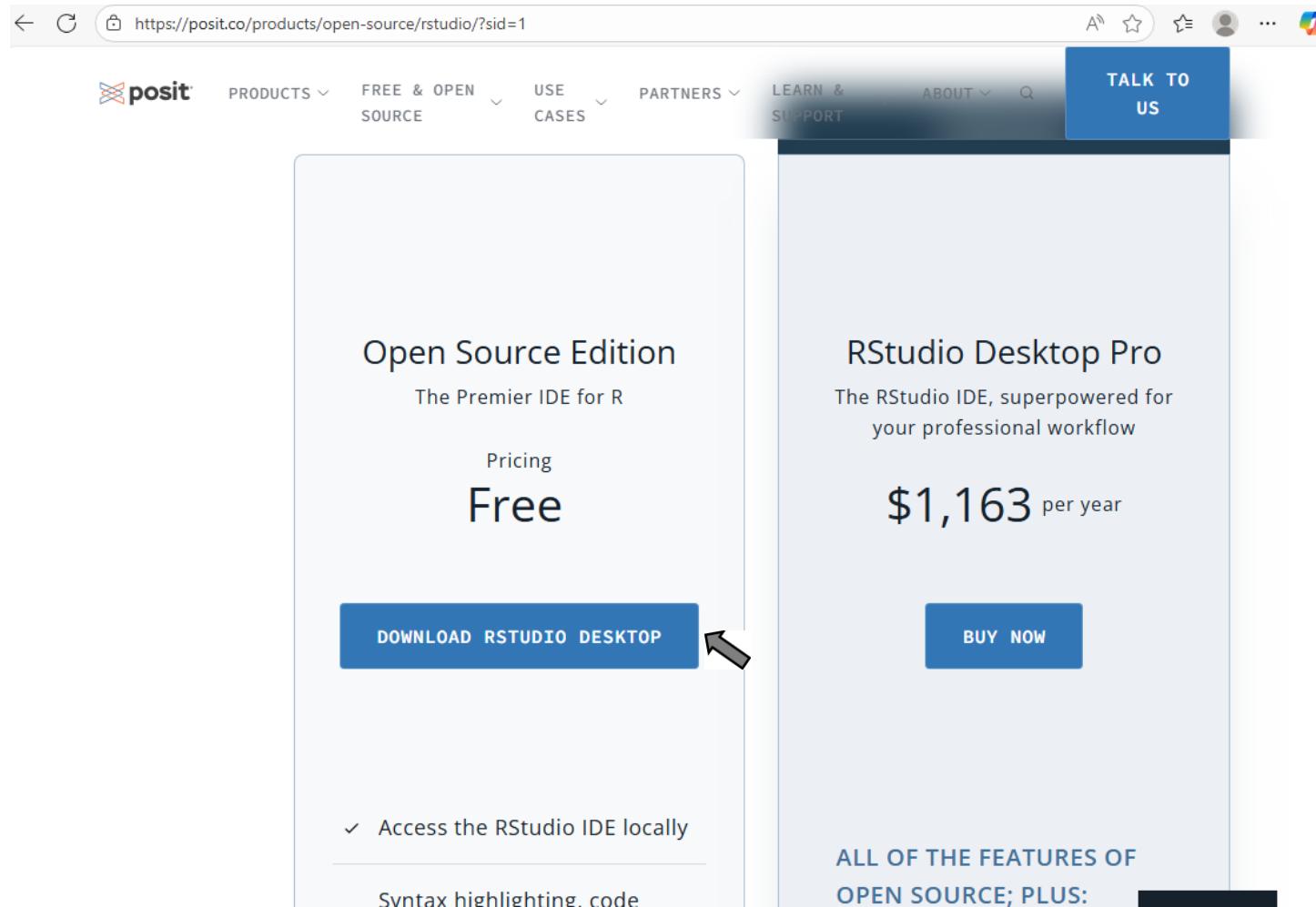
Let's get RStudio

- Download the latest version of RStudio at
<https://posit.co/>



Let's get RStudio

- Choose a “Desktop” version



The screenshot shows the RStudio product page on the posit.co website. The URL in the browser is https://posit.co/products/open-source/rstudio/?sid=1. The page features two main sections: 'Open Source Edition' and 'RStudio Desktop Pro'. The 'Open Source Edition' is described as 'The Premier IDE for R' and is 'Free'. It has a 'Pricing' section and a large blue 'DOWNLOAD RSTUDIO DESKTOP' button. Below the button is a list of features: 'Access the RStudio IDE locally' and 'Syntax highlighting, code'. An arrow points from the text 'Choose a “Desktop” version' to the 'DOWNLOAD RSTUDIO DESKTOP' button. The 'RStudio Desktop Pro' section is described as 'The RStudio IDE, superpowered for your professional workflow' and costs '\$1,163 per year'. It has a 'BUY NOW' button. At the bottom, it says 'ALL OF THE FEATURES OF OPEN SOURCE; PLUS:'.

Open Source Edition
The Premier IDE for R
Pricing
Free

DOWNLOAD RSTUDIO DESKTOP

✓ Access the RStudio IDE locally
Syntax highlighting, code

RStudio Desktop Pro
The RStudio IDE, superpowered for your professional workflow
\$1,163 per year

BUY NOW

ALL OF THE FEATURES OF OPEN SOURCE; PLUS:

Panes of RStudio

The screenshot displays the RStudio interface with the title bar "SIRE516-main - RStudio". The menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The top toolbar contains icons for file operations like Open, Save, and Print, along with "Go to file/function" and "Addins".

The main window is divided into several panes:

- Console pane (leftmost, highlighted with a blue border):** Shows the R startup message and a single command prompt line starting with ">".

```
R version 4.5.2 (2025-10-31) -- "[Not] Part in a Rumble"
Copyright (C) 2025 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Console pane = everything runs here
- Environment pane (top right):** Shows the "Environment" tab selected. It displays a message "Environment is empty" and a search bar.
- Plots pane (bottom right):** Shows tabs for Files, Plots, Packages, Help, Viewer, and Presentation. The "Plots" tab is selected, and there are icons for zooming and exporting.

Panes of RStudio

The screenshot shows the RStudio interface with the title bar "SIRE516-main - RStudio". The menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The top toolbar has icons for file operations like Open, Save, and Print, along with Go to file/function and Addins.

Console pane: Displays the R startup message and a command prompt:

```
R version 4.5.2 (2025-10-31) -- "[Not] Part in a Rumble"
Copyright (C) 2025 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> a <- 0
> |
```

Environment pane: Shows the Global Environment tab with a variable "a" assigned the value 0.

Environment pane annotation: A callout box highlights the Environment pane with the text: "Environment pane = show variables, functions and objects created".

Other panes: The Plots, Packages, Help, Viewer, and Presentation tabs are visible at the bottom of the interface.

Panes of RStudio

The screenshot displays the RStudio interface with several panes visible:

- Console pane:** Shows the R session output, including the R version information, license terms, and a command history starting with `> a <- 0`.
- History pane:** Located in the top right corner, this pane is titled "History pane = recently run commands". It contains a list of R commands that have been run recently, such as setting heatmap dimensions, drawing a heatmap, and defining variable `a`.
- Environment pane:** Located in the top right corner, this pane shows the current environment variables.
- File menu:** Standard file operations like Open, Save, Print, etc.
- Edit menu:** Includes Find, Replace, Cut, Copy, Paste, etc.
- Code menu:** Options for code navigation and refactoring.
- View menu:** Controls the visibility of various panes and toolbars.
- Plots menu:** Options for creating and managing plots.
- Sessions menu:** Management of R sessions and projects.
- Build menu:** Options for building and running R code.
- Debug menu:** Tools for debugging R code.
- Profile menu:** Options for profiling R code.
- Tools menu:** Various tools for R development.
- Help menu:** Help and documentation options.

Panes of RStudio

The screenshot displays the RStudio interface with the following panes:

- Console pane:** Shows the R startup message, license information, natural language support note, contributor information, and a command history snippet.
- Environment pane:** Displays R code for creating a heatmap, including variable definitions and a call to `draw`.
- Files pane:** Lists files in the working directory, including various CSV, XML, and image files.

```
R version 4.5.2 (2025-10-31) -- "[Not] Part in a Rumble"
Copyright (C) 2025 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> a <- 0
>
```

```
heatmap_width = unit(10.5, "cm"),
heatmap_height = unit(8, "cm"),
show_heatmap_legend = FALSE)
complex_heatmap_table
# draw the heat map
draw(complex_heatmap_table, heatmap_legend_list = (list(lg...
heatmap_legend_side ="left")
a <- 0
```

Name	Size	Modified
.Rhistory	22.4 KB	Jan 14, 2026, 5:33 PM
AY2025_assignment	1000.5 KB	Mar 7, 2025, 3:52 PM
Combine.jpeg	178.2 MB	Mar 6, 2025, 8:10 AM
confirmed-cases.csv	49 KB	Mar 29, 2025, 10:41 AM
ec00020.xml	187.3 KB	Oct 27, 2021, 2:42 PM
framingham.csv	36.1 KB	Jan 6, 2025, 1:41 PM
hsa00020.xml	51.2 KB	Jan 6, 2025, 12:17 PM
hsa04110.xml	39 KB	Jan 6, 2025, 12:34 PM
hsa04380.xml	9.9 KB	Mar 7, 2025, 2:10 PM
j.jpeg	1.3 KB	Feb 14, 2025, 3:15 PM
mtcars.csv	1.7 KB	Feb 14, 2025, 3:15 PM
mtcars2.csv	9.7 KB	Jul 26, 2024, 3:05 PM
PIE.jpeg	14.7 KB	Jul 26, 2024, 3:05 PM
PIE.png	9.3 MB	Mar 7, 2025, 2:56 PM
pie.tiff	675.2 KB	Mar 7, 2025, 2:00 PM

Panes of RStudio

SIRE516-main - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Console Terminal x Background Jobs x

R version 4.5.2 (2025-10-31) -- "[Not] Part in a Rumble"
Copyright (C) 2025 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

```
> a <- 0
> plot(x=iris$Sepal.Length,y=iris$Sepal.Width)
> |
```

Environment History Connections

```
heatmap_height = unit(8, "cm"),
show_heatmap_legend = FALSE)
complex_heatmap_table
# draw the heat map
draw(complex_heatmap_table, heatmap_legend_list = (list(lg...
heatmap_legend_side = "left"))
a <- 0
plot(x=iris$Sepal.Length,y=iris$Sepal.Width)
```

Plots pane = recently created plots

A scatter plot titled 'Plots pane = recently created plots'. The x-axis is labeled 'iris\$Sepal.Length' and ranges from 4.5 to 8.0. The y-axis is labeled 'iris\$Sepal.Width' and ranges from 2.0 to 4.0. The plot shows a positive correlation between Sepal.Length and Sepal.Width, with data points clustered into three distinct groups corresponding to the three species in the Iris dataset.

Panes of RStudio

The screenshot displays the RStudio interface with four main panes:

- Console** (leftmost): Shows the R startup message, license information, and a command history including the execution of `plot(x=iris$Sepal.Length, y=iris$Sepal.Width)`.
- Environment** (top right): Displays R objects: `heatmap_height`, `show_heatmap_legend`, `complex_heatmap_table`, `a`, and a plot command.
- Plots** (bottom right): A small preview of a scatter plot.
- Packages** (bottom right, highlighted with a blue box): Shows the "User Library" section of the packages pane, listing various R packages with their descriptions, CRAN/BioC status, and versions.

Package	Description	CRAN/BioC	Version
abind	Combine Multidimensional Arrays	CRAN	1.4-8
ade4	Analysis of Ecological Data: Exploratory	CRAN	1.7-23
adegenet	Exploratory Analysis of Genetic	CRAN	2.1.11
adephylo	Exploratory Analyses for the Ph	https://cloud.r-project.org	1.1-17
ALL	A data package	BioC 3.21	1.50.0
AnnotationDbi	Manipulation of SQLite-based a	BioC 3.21	1.70.0
ape	Analyses of Phylogenetics and I	CRAN	5.8-1
aplot	Decorate a 'ggplot' with Associa	CRAN	0.2.9
AsioHeaders	'Asio' C++ Header Files	https://cloud.r-project.org	1.30.2-1
askpass	Password Entry Utilities for R, G	CRAN	1.2.1
backports	Reimplementations of Function:	CRAN	1.5.0
base64enc	Tools for base64 encoding	CRAN	0.1-3
bayesplot	Plotting for Bayesian Models	https://cloud.r-project.org	1.15.0
beeswarm	The Bee Swarm Plot, an Alterna	CRAN	0.4.0
BH	Boost C++ Header Files	CRAN	1.87.0-1
Biobase	Biobase: Base functions for Bio	BioC 3.21	2.68.0
BiocFileCache	Manage Files Across Sessions	BioC 3.21	2.16.2
BiocGenerics	S4 generic functions used in Bi	BioC 3.21	0.54.1

Panes of RStudio

The screenshot shows the RStudio interface with several panes:

- Console pane:** Displays the R startup message, license information, and a command history including `> a <- 0` and `> plot(x=iris$Sepal.Length,y=iris$Sepal.Width)`.
- Environment pane:** Shows the current environment with objects like `heatmap_height`, `show_heatmap_legend`, `complex_heatmap_table`, and a partial `plot` command.
- Help pane:** A tooltip or callout box highlights the Help pane, which displays the documentation for the `phymltest` function. The documentation includes the function's name, a brief description, usage information, and the source code for the function.

Panes of RStudio

The screenshot displays the RStudio interface with several panes:

- Top Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for New Project, Open Project, Save, Undo, Redo, Go to file/function, Addins, and a search icon.
- Console Tab:** Shows the R startup message and a command history:

```
R version 4.5.2 (2025-10-31) -- "[Not] Part in a Rumble"
Copyright (C) 2025 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> a <- 0
> plot(x=iris$Sepal.Length,y=iris$Sepal.Width)
> |
```
- Environment Tab:** Shows R code for creating a heatmap:

```
heatmap_height = unit(8, "cm"),
show_heatmap_legend = FALSE)
complex_heatmap_table
# draw the heat map
draw(complex_heatmap_table,heatmap_legend_list = (list(lg...
heatmap_legend_side ="left")
a <- 0
plot(x=iris$Sepal.Length,y=iris$Sepal.Width)
```
- Help Tab:** Shows the help page for the `phymltest` function:

R: Fits a Bunch of Models with PhyML

Description

This function calls PhyML and fits successively 28 models of DNA evolution. The results are saved on disk, as PhyML usually does, and returned in **R** as a vector with the log-likelihood value of each model.

Usage

```
phymltest(seqfile, format = "interleaved", itree = NULL,
          exclude = NULL, execname = NULL, append = TRUE)
## S3 method for class 'phymltest'
print(x, ...)
## S3 method for class 'phymltest'
summary(object, ...)
## S3 method for class 'phymltest'
plot(x, main = NULL, col = "blue", ...)
```
- Documentation Tab:** Shows the R Documentation page for the `phymltest` function.

Source Panes

- Commands entered in the Console Pane will be immediately executed.
- To rerun the commands, they must be retyped in the Console Pane or reloaded from the History Pane
- Source pane
 - Write commands in this pane to be saved and executed later.
 - Develop R script here

Open Source Pane

The screenshot shows the RStudio interface with the title bar "SIRE516-main - RStudio". The left pane is the "Source" pane, which displays the R code for the Iris dataset and its visualization. The right pane is the "Open Source" pane, which shows the documentation for the `phymltest` function from the `ape` package. A red arrow points to the "R Script" icon in the Source pane's toolbar.

SIRE516-main - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

R Script Ctrl+Shift+N

Quarto Document... Quarto Presentation... R Notebook R Markdown... Shiny Web App... Plumber API... Text File C++ File Python Script SQL Script Stan File D3 Script R Sweave R HTML R Documentation...

(2025-10-31) -- "[Not] Part in a Rumble"
025 The R Foundation for Statistical Computing
4-pc-linux-gnu

are and comes with ABSOLUTELY NO WARRANTY.
to redistribute it under certain conditions.
' or 'licence()' for distribution details.

age support but running in an English locale

ative project with many contributors.

ors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.

Type 'q()' to quit R.

```
> a <- 0
> plot(x=iris$Sepal.Length,y=iris$Sepal.Width)
>
```

Environment History Connections

To Console To Source

```
heatmap_height = unit(8, "cm"),
show_heatmap_legend = FALSE)
complex_heatmap_table
# draw the heat map
draw(complex_heatmap_table, heatmap_legend_list = (list(lg...
heatmap_legend_side ="left")
a <- 0
plot(x=iris$Sepal.Length,y=iris$Sepal.Width)
```

Files Plots Packages Help Viewer Presentation

R: Fits a Bunch of Models with PhyML Find in Topic

phymltest {ape}

Fits a Bunch of Models with PhyML

Description

This function calls PhyML and fits successively 28 models of DNA evolution. The results are saved on disk, as PhyML usually does, and returned in **R** as a vector with the log-likelihood value of each model.

Usage

```
phymltest(seqfile, format = "interleaved", itree = NULL,
exclude = NULL, execname = NULL, append = TRUE)
## S3 method for class 'phymltest'
print(x, ...)
## S3 method for class 'phymltest'
summary(object, ...)
## S3 method for class 'phymltest'
plot(x, main = NULL, col = "blue", ...)
```

Open Source Pane

The screenshot shows the RStudio interface with the following components:

- Source pane:** Displays the R script code for generating a heatmap.
- Environment pane:** Shows the variables and objects available in the current R session.
- Help pane:** Provides documentation for the `phylmtest` function.

Source pane content:

```
1Source pane = write R script here to be run later
```

Environment pane content:

```
heatmap_height = unit(8, "cm"),
show_heatmap_legend = FALSE)
complex_heatmap_table
# draw the heat map
draw(complex_heatmap_table, heatmap_legend_list = (list(lg...
heatmap_legend_side ="left")
a <- 0
plot(x=iris$Sepal.Length,y=iris$Sepal.Width)
```

Help pane content:

Fits a Bunch of Models with PhyML

Description

This function calls PhyML and fits successively 28 models of DNA evolution. The results are saved on disk, as PhyML usually does, and returned in R as a vector with the log-likelihood value of each model.

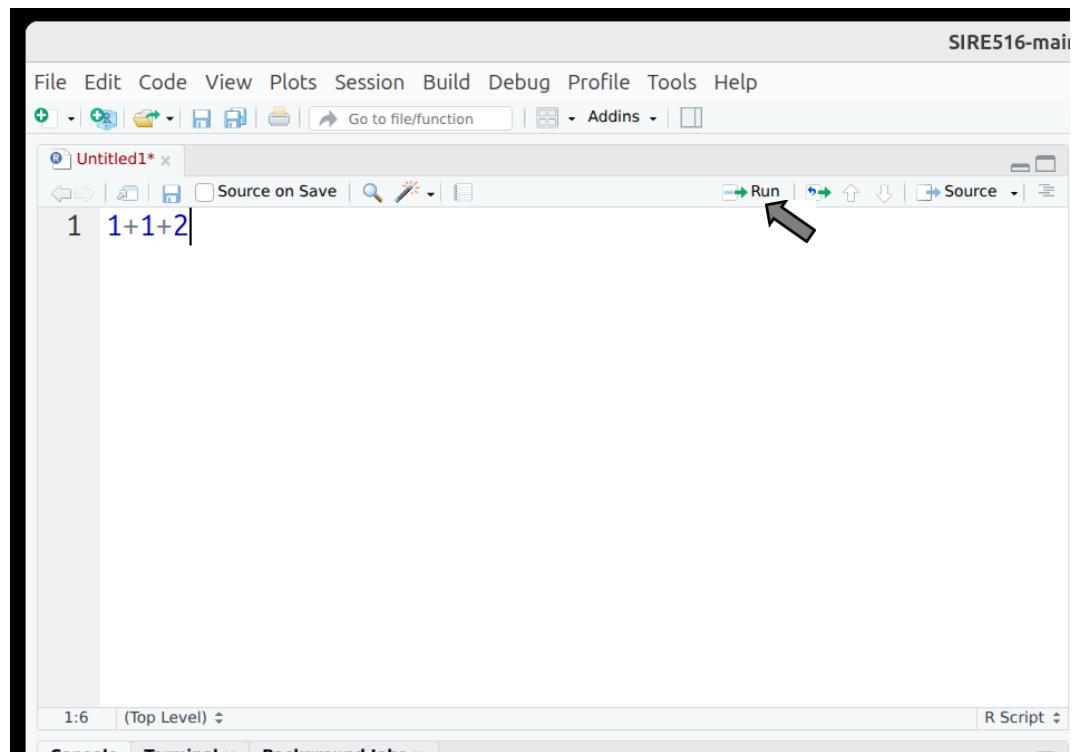
Usage

```
phylmtest(seqfile, format = "interleaved", itree = NULL,
          exclude = NULL, execname = NULL, append = TRUE)
## S3 method for class 'phylmtest'
print(x, ...)
## S3 method for class 'phylmtest'
summary(object, ...)
## S3 method for class 'phylmtest'
plot(x, main = NULL, col = "blue", ...)
```

Run scripts in Source Pane

Run a single line of the script

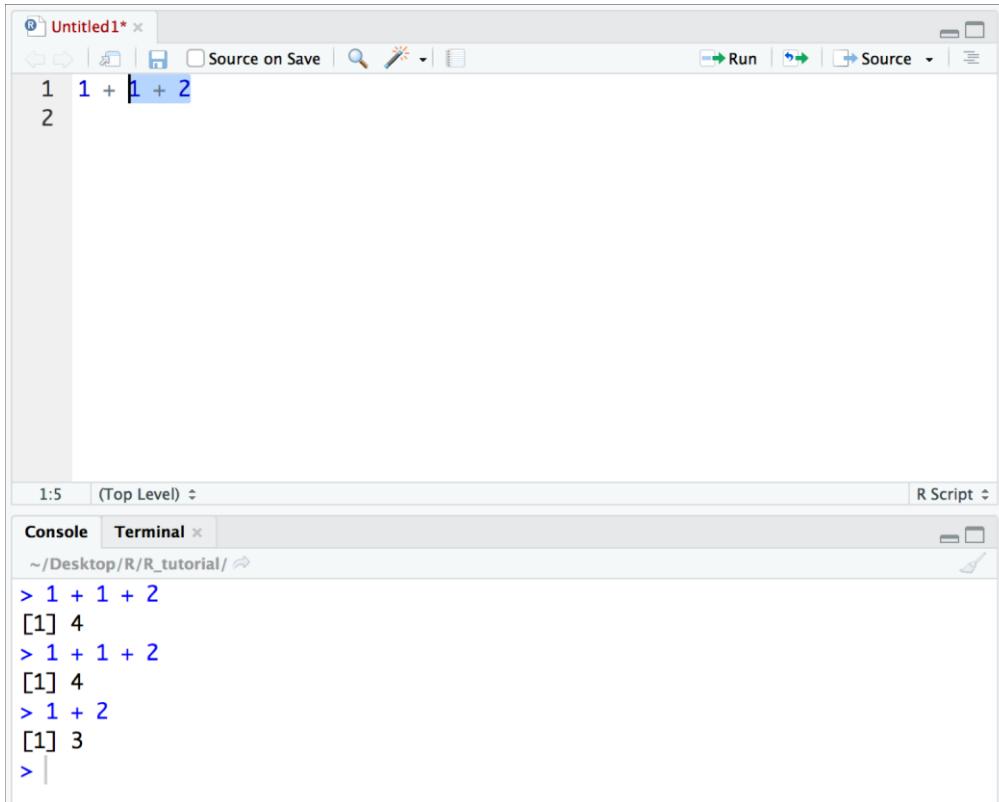
- Leave a cursor at a line
- Click  or
- Ctrl + Enter or
- Mac: Command + Enter



Run scripts in Source Pane

Run a part of the script

- Highlight the part
- Click  or
- Ctrl + Enter or
- Mac: Command + Enter



The screenshot shows the RStudio interface. The Source pane at the top contains the R code:

```
1 1 + 1 + 2
2
```

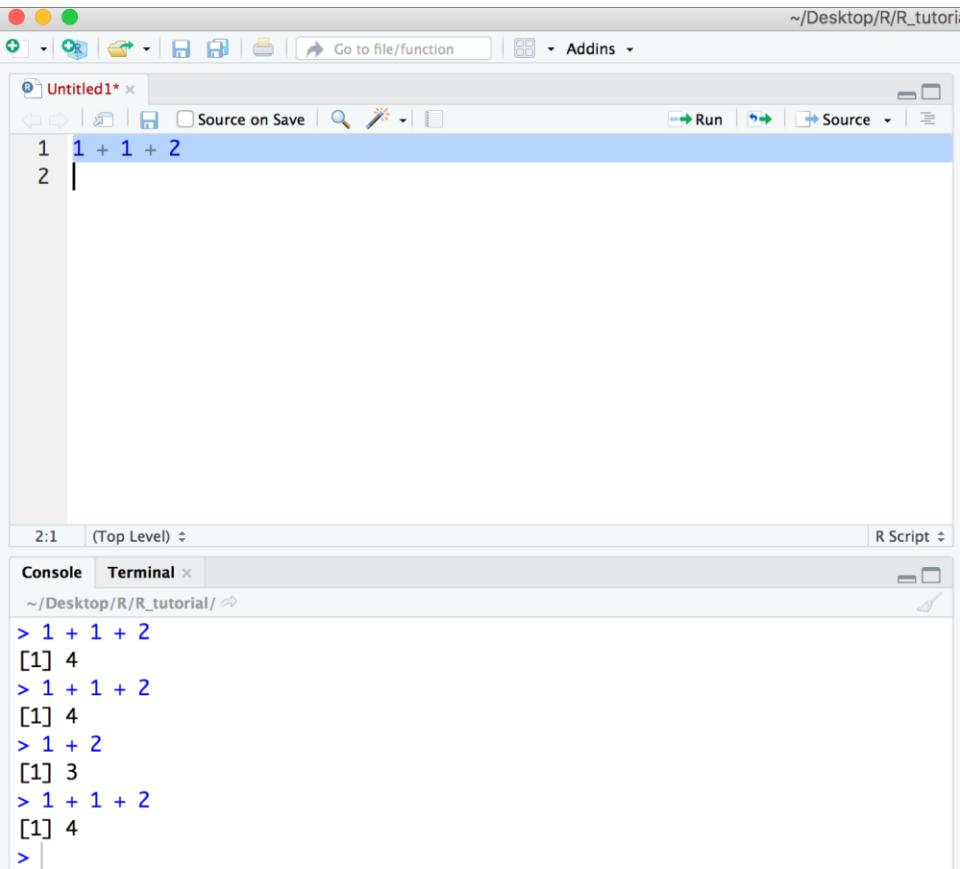
The line `1 + 1 + 2` is highlighted with a blue selection bar. Below the Source pane is the Console pane, which displays the output of running the script:

```
1:5 (Top Level) > R Script
Console Terminal ~~/Desktop/R/R_tutorial/
> 1 + 1 + 2
[1] 4
> 1 + 1 + 2
[1] 4
> 1 + 2
[1] 3
>
```

Run scripts in Source Pane

Run all lines of the script

- Click the source pane
- Ctrl/Command + A
- Click  or
- Ctrl/Command + Enter



The screenshot shows the RStudio interface. The top panel is the Source pane, titled "Untitled1*", containing the R code `1 + 1 + 2`. The bottom panel is the Console pane, titled "2:1 (Top Level)", showing the output of running the script:

```
> 1 + 1 + 2
[1] 4
> 1 + 1 + 2
[1] 4
> 1 + 2
[1] 3
> 1 + 1 + 2
[1] 4
>
```

Useful keyboard shortcuts

- Ctrl + S = Save the script
- Ctrl + L = Clear the console pane
- Ctrl + Z = Undo and Ctrl + Y = Redo
- Ctrl + X =Cut, Ctrl + C = Copy, Ctrl + V =Paste
- **Tab = ***Autofill*****
- **↑ = ***Recall previously executed command*****
- Ctrl + Shift + F10 = Restart R session
- Ctrl + Q = Quit R

More shortcuts: Tools → Keyboard Shortcuts Help

Practical 0: Packages

- R is powerful and versatile because of “Packages”
- “Packages” are codes/scripts created by users and shared to the R community
- 23,029 available packages deposited at CRAN (As of January 2025)

Practical: Packages

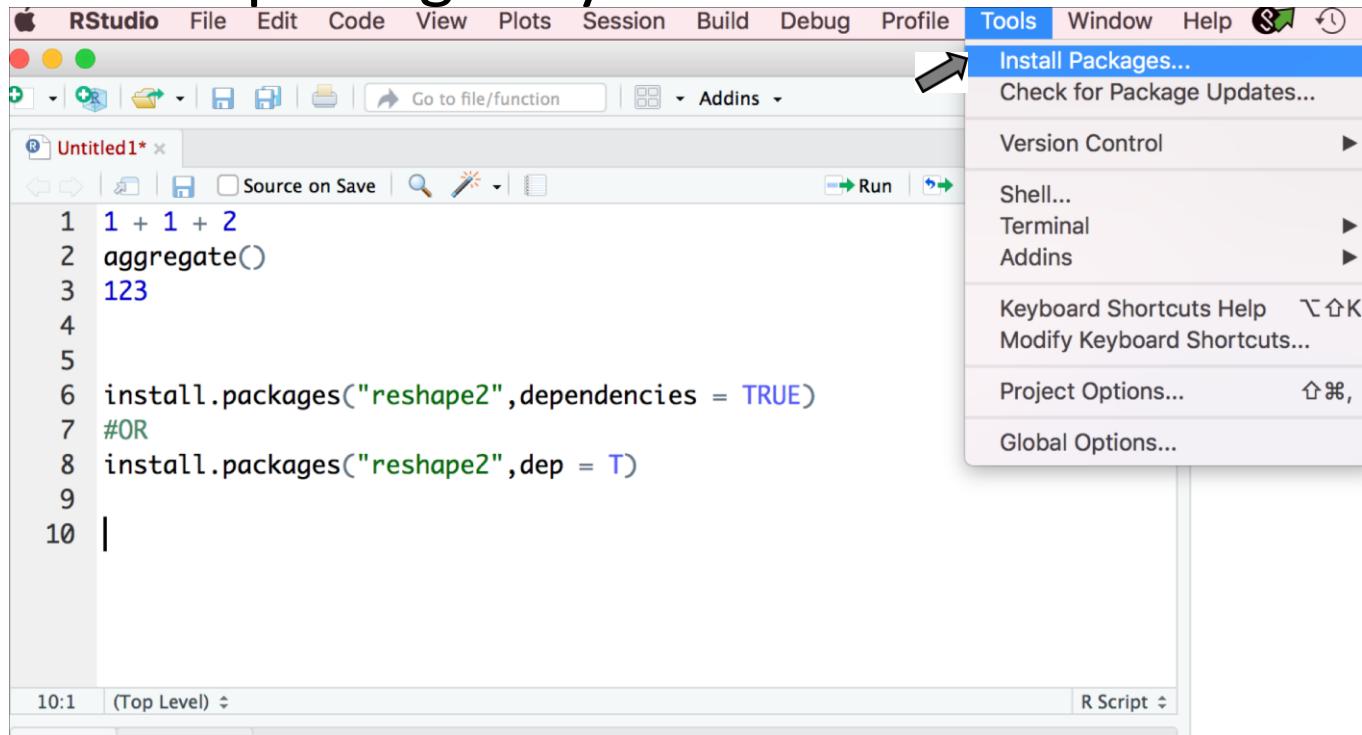
- To install packages by a command line

```
install.packages("reshape2",dependencies = TRUE)  
#OR  
install.packages("reshape2",dep = T)
```

- `install.packages("name of the package in quotation marks", dependencies = TRUE)`
- “`dependencies = TRUE`” means other packages required by this package will also be downloaded

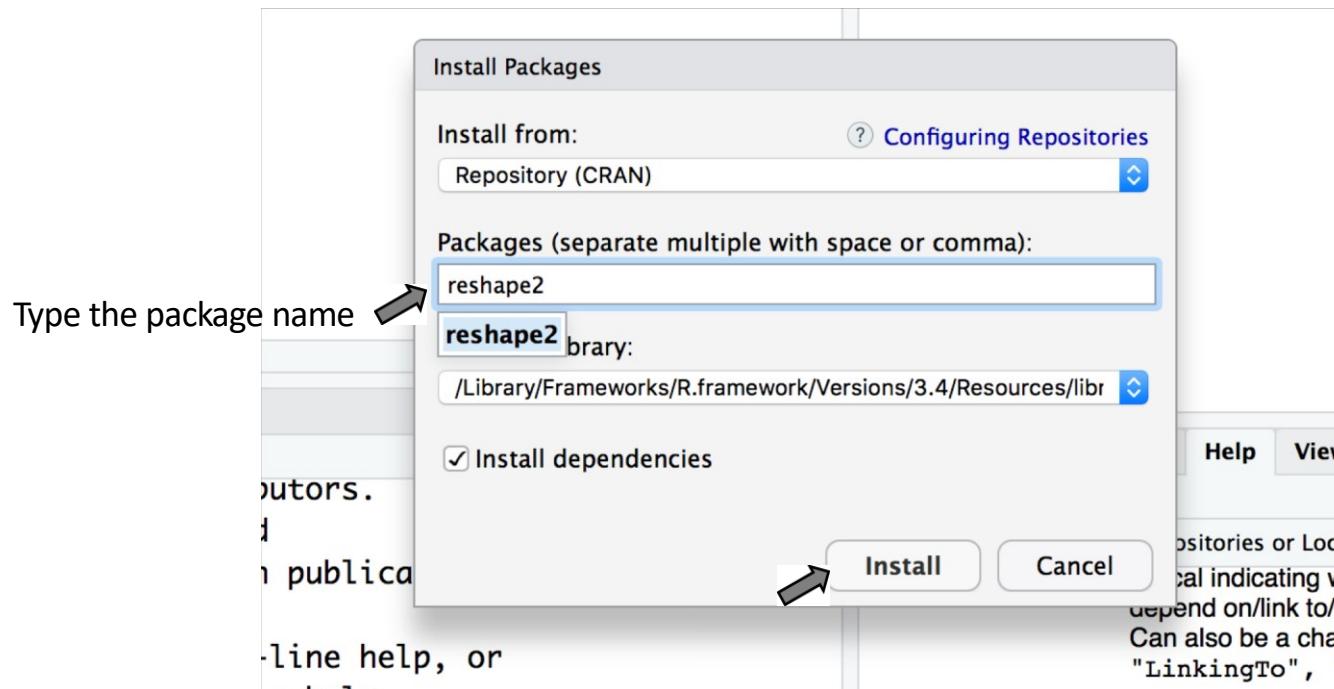
Practical: Packages

- To install packages by RStudio



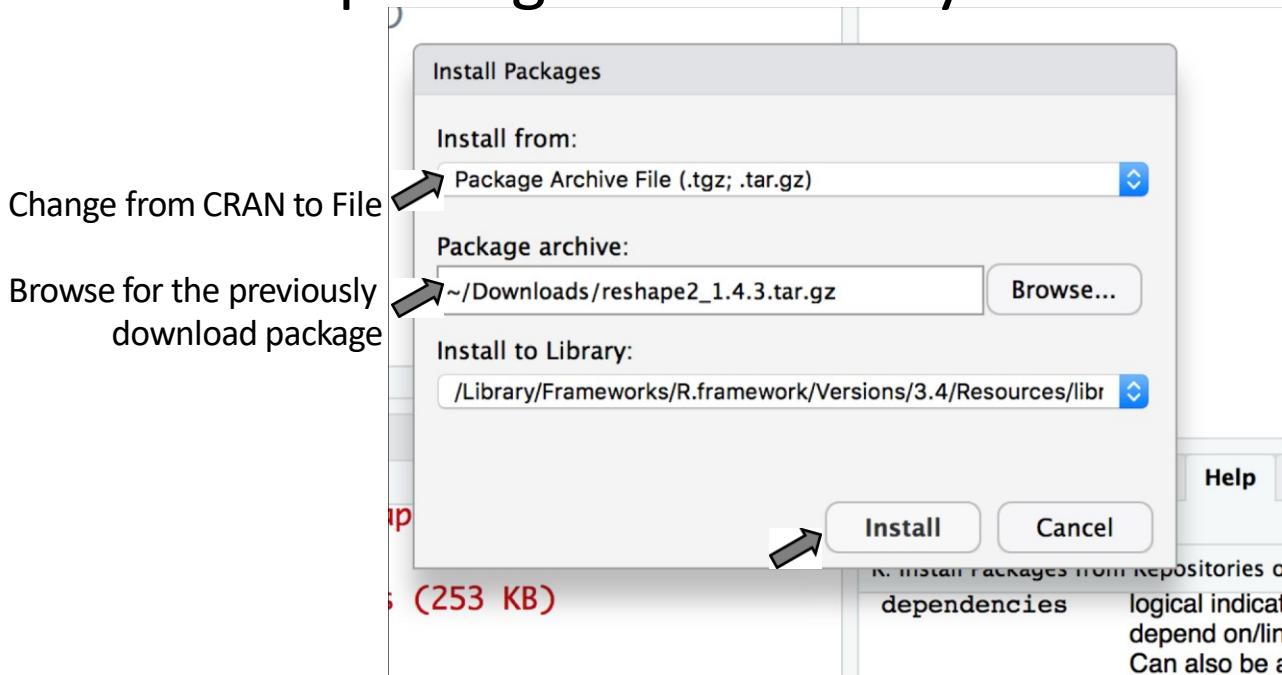
Practical: Packages

- To install packages by RStudio



Practical: Packages

- To install packages “offline” by RStudio

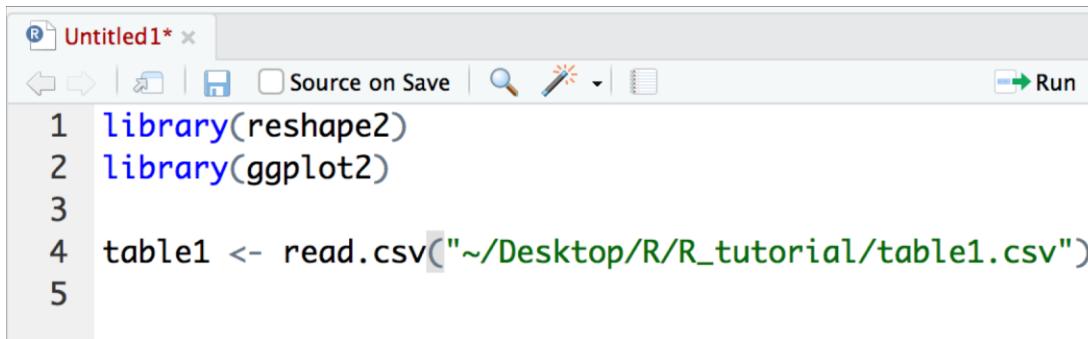


Practical: Packages

- To invoke the package

```
library(reshape2) #In your script  
#OR  
require(rehape2) #In the function/package
```

- Usually at the top of your script/code



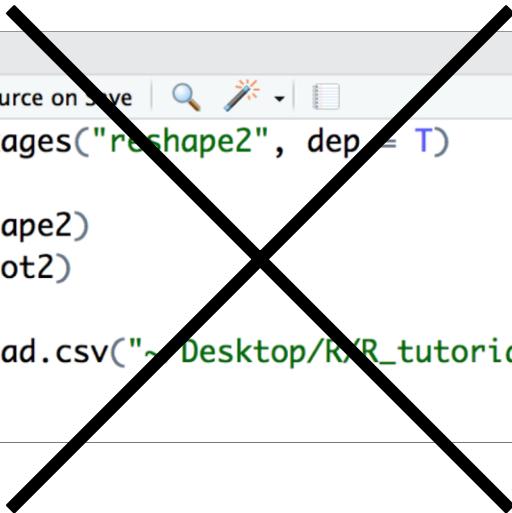
A screenshot of the RStudio interface showing an R script titled "Untitled1". The script contains the following code:

```
1 library(reshape2)  
2 library(ggplot2)  
3  
4 table1 <- read.csv("~/Desktop/R/R_tutorial/table1.csv")  
5
```

The code is highlighted with syntax coloring, and the RStudio toolbar is visible at the top.

Practical: Packages

- DO NOT include “install.packages()” in your script/code
- This will result in reinstalling packages every time the script is run



```
R Untitled1* x
Source on Save | Run | Save
1 install.packages("reshape2", dep = T)
2
3 library(reshape2)
4 library(ggplot2)
5
6 table1 <- read.csv("~/Desktop/R/R_tutorial/table1.csv")
7 |
```

Practical: Packages

- Your turn:
 - Install and invoke the following packages:
 - reshape2
 - lubridate
 - MASS
 - car
 - Exact
 - hflights
 - readr
 - readxl

Programming Concepts in R

FILE:Programming_Concepts.R

Before we start...

- R scripts for demonstration are prepared
 - Run scripts and see what happens in your machine
 - Tell us if you see any strange or different results
 - Feel free to copy and modify the scripts for your own work

Calculator

Basic Mathematical Operations

```
1 - #####Basic Mathematical Operations pt 1#####
2 1 + 2 #Addition
3 1 - 2 #Substraction
4 1 / 2 #Division
5 1 * 2 #Multiplication
6 2 ^ 5 #Power
7 sqrt(4) #Square root
8 5 %/ 2 #Integer division
9 5 %% 2 #Modulo
```

- TIP: '#' = comment sign; Nothing after # to the end of the line will be run!!!

Basic Mathematical Operations

```
11 - #####Basic Mathematical Operations pt 2#####
12 log10(100) #Logarithm base 10
13 log(100, base = 10) #Logarithm base 10
14 log(100) #Natural logarithm
15 exp(1) #Exponential i.e. e^1
16 round(2.111, digits = 1) #Rounding
17 floor(2.111) #Round down to integer
18 ceiling(2.111) #Round up to integer
19 abs(-123) #Absolute value
```

Basic Logical Operations

```
21 - #####Basic Logical Operations pt 1#####
22 20 == 18 #Comparing whether 'left' equals to 'right.'
23
24 20 = 18 #This is wrong. Try and see what happens.
25
26 20 != 18 #Comparing whether 'left' does not equal to 'right.'
27 20 < 18 #Comparing whether 'left' is less than 'right.'
28 20 > 18 #Comparing whether 'left' is greater than 'right.'
29 20 <= 18 #Comparing whether 'left' is less than or equal to 'right.'
30 20 >= 18 #Comparing whether 'left' is greater than or equal to 'right.'
31
32 20 =~ 18 #This is wrong. Try and see what happens.
33 20 =>~ 18 #This is wrong. Try and see what happens.
```

Basic Logical Operations

```
35 - #####Basic Logical Operations pt 2#####
36 (20 > 18) & (20 < 18) #'AND' operation
37 (20 > 18) | (20 < 18) #'OR' operation
38 !(20 < 18) #'NOT' operation
```

Variable Assignment

Variable assignment

```
40 ####Variable assignment pt 1####
41 (20 > 18) & (20 < 18)
42 #Versus
43 a <- (20 > 18)
44 b <- (20 < 18)
45 c <- a & b
46 print(c) #'print()' is to show the values of the variable
47
48 d <- (20 == 18)
49 e <- c | d #Versus ((20 > 18) & (20 < 18)) | (20 == 18)
50 print(e)
```

- Critical concept in programming
- Save results/outputs for later
- Make your code more legible
- More legible = Easier for debugging

Variable assignment

- How to assign values to variables

```
52 - #####Variable assignment pt 2#####
53 a = 1 #Generic: Assign the 'right' value to the 'left' variable
54
55 a <- 1 #R: Assign the 'right' value to the 'left' variable
56 1 -> a #R: Assign the 'left' value to the 'right' variable
57 a <- 1 -> b #R: Multiple assignments
58 a <- b <- 1 #R: Multiple assignments
```

- ‘=’ and ‘<-’ are mostly interchangeable
- Personally, I prefer ‘<-’ to avoid the confusion between ‘=’ and ‘==’

Variable assignment

- Naming your variables
 - Start with alphabet or ‘.’
 - No space or special character in the name except ‘.’ and ‘_’
 - Be careful! R is case-sensitive

```
60 ######Variable assignment pt 3#####
61 a <- 123 #Correct
62 a1 <- 123 #Correct
63 1a ~~~~ 123 #Incorrect
64 a 1 ~~~~ 123 #Incorrect
65 a_1 <- 123 #Correct
66 a.1 <- 123 #Correct #Different meaning in Java or Python
67 a! ~~~~ 123 #Incorrect
68 a? ~~~~ 123 #Incorrect
69 .a <- 123 #Correct
70 print(a1) #R is case-sensitive
71 print(A1) #R is case-sensiitive
```

Types of Data

Basic types of data

```
73 - #####Types of data pt 1#####
74 a <- 2 #numeric
75 b <- TRUE #logical
76 c <- "Hello, World!" #character #Must be in " " or '
77 c <- Hello, World! #Try and see
78 d <- NA #missing (logical)
79 e <- NaN #'Not a Number' = ill-defined e.g. 0/0 (logical)
80
81 typeof(a) #Check the 'R internal' type of data by 'typeof()'
82 class(a) #Check the 'customized' type of data by 'class()'
83 #class() is used more often.
84
85 is.logical(b) #'is.logical()' is for checking if the data is 'logical'
86 is.logical(d)
87 is.na(b) #'is.na()' is an essential function for managing missing data.
88 is.na(d)
89 #is.nan(); is.numeric(); is.character() and others
```

Complex types of data

- More complex types of data or ‘objects’
- Containing multiple elements of basic data types
(i.e. They are still based on basic types of data).
- Some types were developed for certain applications
(e.g. spatial objects, survival objects)
- They might be converted to basic types of data.

Factor

- Recoding ‘categorical’ or ‘ordinal’ values to calculable values
 - Character + Numeric
- Example: Categorical variables in regression analysis

```
90 - #####Types of data pt 2#####
91 a <- "good"
92 class(a)
93 typeof(a)
94 af <- factor(a, levels = c("terrible","bad","neutral","good","excellent"))
95 print(af)
96 class(af)
97 typeof(af)
98 as.numeric(af) #Convert to numeric, i.e. extracting numerical elements
99 as.character(af) #Convert to character, i.e. extracting character elements
```

Date' and 'Time'

- Recoding texts of 'date', 'time' or 'datetime' data to calculable values
- Example: Used in time series analyses, survival analyses

```
101 - #####Types of data pt 3#####
102 library(lubridate)
103
104 a <- "27-3-2018"
105 class(a)
106 typeof(a)
107 ad <- dmy(a) #function in lubridate package converting 'character' to 'date'
108 print(ad)
109 class(ad)
110 typeof(ad)
111 as.numeric(ad) #Convert to numeric (i.e. Days since 1-1-1970)
112 as.character(ad) #Convert to character, i.e. extracting character elements
```

Data type conversion

- Frequently used commands for data type conversion

```
114 ####Conversion#####
115 a <- 1
116 print(a)
117 as.character(a) #Convert to character
118 as.factor(a) #Convert to factor; cannot change levels or labels
119 factor(a, levels = c(3,2,1)) #Specify levels
120 factor(a, levels = c(1,2,3), labels = c("one","two","three")) #Specify labels
121
122 b <- "111"
123 as.numeric(b)
124 c <- "1,111"
125 as.numeric(c) #CAUTION: Data conversion may introduce missing data
126 d <- "1111"
127 as.numeric(d)
```

Data Structure
(Data container)

Vector

- One-dimensional container
- Every member in the vector must be the same type
- `c()` to construct a vector

```
114 ####Vector pt 1####
115 A <- c(1,2,3,4)
116 print(A)
117 class(A)
118
119 B <- c("a","b","c","d")
120 print(B)
121 class(B)
122
123 C <- c(TRUE, FALSE, T, F)
124 print(C)
125 class(C)
```

```
127 ####Vector pt 2####
128 D <- c(1,NA,3,0/0)
129 print(D)
130 class(D)
131
132 E <- c(1,TRUE,0/0,"a")
133 print(E)
134 class(E)
```

Matrix

- Two-dimensional container
- Every member in the matrix must be the same type
- `matrix()` to construct a matrix

```
136 ######Matrix pt 1#####
137 a <- 1:8 #a <- c(1,2,3,4,5,6,7,8) or a <- seq(from=1,to=8,by=1)
138 print(a) #vector of data
139 A <- matrix(data = a,nrow = 4)
140 #load 'a' vector into a matrix
141 #nrow' indicates number of rows
142 print(A)
143
144 B <- matrix(data = a,ncol = 4) #'ncol' indicates number of columns
145 print(B)
146
147 C <- matrix(data = a,ncol = 4, byrow = T)
148 #'byrow' fill values by row
149 # Default is by column (i.e. byrow = F)
150 print(C)
```

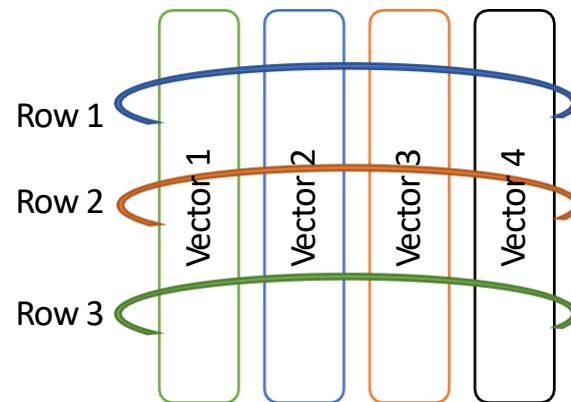
Matrix

- Two-dimensional container
- Every member in the matrix must be the same type
- `matrix()` to construct a matrix

```
152 - #####Matrix pt 2#####
153 D <- matrix(data = c(1:6,NA,T),ncol = 4, byrow = T)
154 print(D) #Look at the last member of the matrix
155
156 E <- matrix(data = c(1:6,T,"a"),ncol = 4, byrow = T)
157 print(E) #Look at the type of data in the matrix
```

Data frame

- Two-dimensional container
- In the “data handling” section (Week2), we will mostly use data frames
- Every member in each “column” of the data frame must be the same type
- Each row can contain different types of data
- Imagine multiple vectors with identical lengths with each member at the same position bound together as a ‘row.’



Data frame

- `data.frame()` to construct a data frame

```
159 ####Data frame pt 1####
160 A <- data.frame( name = c("A", "B", "C"), #Vector 1 assigned as 'name' column
161           weight = c(50,60,55), #Vector 2 assigned as 'weight' column
162           height = c(150,166,162)) #Vector 3 assigned as 'height' column
163 print(A)
164
165 A2 <- data.frame(c("A", "B", "C"), #No column names assignment
166                   c(50,60,55),
167                   c(150,166,162))
168 print(A2)
169
170 B <- data.frame( name = c("B", "C"),
171           weight = c(50,60,55),
172           height = c(150,166,162)) #Try and see what happens
```

Data frame

- `data.frame()` to construct a data frame

```
175 ####Data frame pt 2####
176 C <- data.frame( name = c("A", "B", "C"),
177                   weight = c(50, 60, "Not measured"),
178                   height = c(150, 166, 162))
179 print(C) #Try and see
180 A$weight 'A$weight' is for calling the 'weight' column of 'A' data frame
181 #We will get into how to access data in the data frame in the later session
182 class(A$weight)
183
184 C$weight
185 class(C$weight)
```

List

- Multi-dimensional container
- Each member in the list can be any type of data, or even data container (i.e. a data frame in a list > two dimensions)
- No restriction of length for each member
- Used as a container of results of analysis (e.g. regression results)

List

- `list()` to construct a list

```
188 ####List####
189 a <- c(T,F)
190 b <- 1:6
191 c <- c("A","E","I","O","U")
192
193 A <- list(a,b,c) #Combine vectors as a list
194 print(A)
195
196 A2 <- list(col1 = a, col2 = b, col3 = c) #'name' can be assigned to each member
197 print(A2)
198
199 d <- data.frame( name = c("A","B","C"),
200                               weight = c(50,60,55),
201                               height = c(150,166,162))
202 B <- list(A,d) #list and data frame in the list
203 print(B)
```

Practical 1

Practical 1: Data type conversion

- Numerical data are sometimes created or imported to R as ‘factor’ data.
- Incorrectly using factor data can be problematic

```
1 C <- data.frame( name = c("A", "B", "C"),
2                   weight = c(50, 60, "Not measured"),
3                   height = c(150, 166, 162))
4 weight <- C$weight
5 #Convert weight from kilograms to pounds
6 #1 Kilo = 2.2 pounds
7 weight * 2.2 #Try and see
```

Practical 1: Data type conversion

1. Convert 'weight' to numerical data
2. Convert 'weight' to pound (1 kilo ~ 2.2 pound)
 - HINT1: Correct answer = c(110, 132, NA)
 - HINT2: as.numeric(); as.character() for data type conversion

```
1 C <- data.frame( name = c("A", "B", "C"),
2                   weight = c(50, 60, "Not measured"),
3                   height = c(150, 166, 162))
4 weight <- C$weight
5 #Convert weight from kilograms to pounds
6 #1 Kilo = 2.2 pounds
7 weight * 2.2 #Try and see
```

Flow Control

Curly brackets { }

- ‘if-else’, ‘for’, ‘while’ and ‘function’

```
command(conditions or values) { do  
    something  
    do something do something  
}
```

If and Else

- Modify the behavior of the code based on the input
- Evaluate the input -> Select what to do -> Generate the output
- Note: If-Else must address all possible conditions

```
220 #####If and else#####
221 A <- 1| #Assign the input here
222
223 if(!is.numeric(A)){
224   print("A is not a number")
225 }else if(is.nan(A)){
226   print("A is not a number")
227 }else if(A %% 2 == 0){
228   print("A is even")
229 }else if(A %% 2 == 1){
230   print("A is odd")
231 }else{
232   print("A is not an integer")
233 }
234
235 #Try changing A to 2, 1.1, NA and 0/0
236
```

For loop

- Repeating an operations with changing inputs
- NOTE: ‘For loop’ in R is slow (try row/column operations and apply())

```
220 - #####For loop#####
221 A <- c(1,2,3,4,5,6,7)
222 for(i in A){
223   #'i' is a temporary variable with its value = each member of A
224   #Reassignment of 'i' occurred at every new round
225   print(i)
226 }
227
228 B <- letters #letters and LETTERS are built-in vectors containing alphabets
229 for(j in B){
230   #'j' is a temporary variable with its value = each member of A
231   #Reassignment of 'j' occurred at every new round
232   print(j)
233 }
```

While loop

- Repeating an operations until the ‘breaking condition’ is triggered
- CAUTION: ‘Infinite loop’ = the ‘breaking condition’ is never triggered
- NOTE: ‘While loop’ in R is slow

```
222 - #####While loop#####
223 i <- 1
224
225 - while(i < 100){ #Breaking/Exiting condition
226   print(i) #Do something here
227   i <- i + 1 #DO NOT forget to change breaking condition!!!
228 }
```

Practical 2

Practical 2: For loop

- ‘paste()’ is a function that combines two or more text data into one
- Try `paste("I", "love", "you.")`
- ‘paste0()’ is a function that combines two or more text data into one without adding a space between texts
- Try `paste0("I", "love", "you.")`

Practical 2: For loop

1. Our input is 2005:2018 (i.e. 2005, 2006,...,2017, 2018)
2. We will use the for loop and the ‘paste’ function to generate the following sentences:
 - This year is 2005.
 - This year is 2006.
 - This year is 2007.
 - ...
 - ...
 - This year is 2018.

```
1 for(k in 2005:2018){  
2   #Do something with k  
3   #print(something)  
4 }
```

Function

Function

- You can customize your own function
- Function calls multiple lines of code in a simple command
- NOTE: “Package” is a bundle of functions, manual/documentations and sample datasets

```
220 #####If and else#####
221 A <- 1 #Assign the input here
222
223 if(!is.numeric(A)){
224   print("A is not a number")
225 }else if(is.nan(A)){
226   print("A is not a number")
227 }else if(A %% 2 == 0){
228   print("A is even")
229 }else if(A %% 2 == 1){
230   print("A is odd")
231 }else{
232   print("A is not an integer")
233 }
234
235 #Try changing A to 2, 1.1, NA and 0/0
236
```

Function

- Function should be defined at the beginning of the code after loading packages

```
230 - #####Function pt 1#####
231 library(reshape2) #Nothing to do with the code
232 #Just demo how the code should look like
233 odd_even <- function(A){ #Define the 'odd_even' function
234   if(!is.numeric(A) | is.nan(A)){#Debugged for NaN
235     result <- "Your input is not a number"
236   }else if(A %% 2 == 0){
237     result <- "Your input is even"
238   }else if(A %% 2 == 1){
239     result <- "Your input is odd"
240   }else{
241     result <- "Your input is not an integer"
242   }
243   return(result)
244 }
245
246 odd_even(A = 12) #Call the function by name
247 odd_even(0/0)
```

Function

- Function can take multiple arguments
- Default value may be defined

```
249 - #####Function pt 2#####
250 - testFn <- function(A,B,C = 2){#C is 2 by default
251   result <- (A+B)*C
252   return(result)
253 }
254
255 testFn(A = 1, B = 2) #C is 2 by default
256 testFn(A = 2, B = 3, C = 4)
257 testFn(0,4,1) #A = 0, B = 4, C = 1 -> Assign by the order of arguments
258 testFn(C = 0,B = 4, A = 1) #Specifically assign value to A,B,C
```

Help!!!

Help!!!

- General programming concepts in R are similar to other languages
- Tough parts are ‘function’
- Find the right function and learn to use it can be difficult

```
260 - #####Help#####
261 help("shapiro.test")
262 ?shapiro.test
263 ??shapiro.test
```

Prepare your workspace

Working directory

- `getwd()` = the current working directory
 - Anything saved without a specified path will be here
- `setwd()` = change to the new working directory
 - Organizing the location in which files will be read and written

Before running your codes

- Restart “R session”: Session -> Restart R
- Clear the environment
- Check your working directory



Troubleshooting

- Command `sessioninfo::session_info()` is useful for consulting your problems and errors with the community.

```
> sessioninfo::session_info()
- Session info -----
  setting  value
  version   R version 4.4.2 (2024-10-31)
  os        Ubuntu 24.04.1 LTS
  system    x86_64, linux-gnu
  ui        RStudio
  language (EN)
  collate   en_US.UTF-8
  ctype     en_US.UTF-8
  tz        Asia/Bangkok
  date      2025-01-07
  rstudio   2023.06.0+421 Mountain Hydrangea (desktop)
  pandoc   NA

- Packages -----
  package    * version date (UTC) lib source
  cli         3.6.3   2024-06-21 [1] CRAN (R 4.4.2)
  rstudioapi  0.17.1  2024-10-22 [1] CRAN (R 4.4.2)
  sessioninfo 1.2.2   2021-12-06 [1] CRAN (R 4.4.2)

[1] /home/dumrong/R/x86_64-pc-linux-gnu-library/4.4
[2] /usr/local/lib/R/site-library
[3] /usr/lib/R/site-library
[4] /usr/lib/R/library
```

In-class assignment

Solving quadratic equation

- Quadratic equation
 - $ax^2 + bx + c = 0$
- Quadratic formula is the solution of the quadratic equation
 - $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

Solving quadratic equation

1. Create a function ‘quardForm()’ that takes input a, b, c from a quadratic equation to solve for x
2. Test your function with the following equations:
 - $x^2 + 4x - 21 = 0$
 - $10x^2 + 13x - 3 = 0$
 - $x^2 - 25 = 0$
 - $x^2 + 4 = 0$

Note: Your function should work if the answers are “real” numbers.

Solving quadratic equation

- HINT: quadratic formula needs to return two values

```
1 quadForm <- function(A,B,C){  
2   #answer1 <- ???  
3   #answer2 <- ???  
4   return(c(answer1, answer2))  
5 }
```