Introduction to R

Dr. Anup Kumar Sharma
Department of Mathematics
NIT-Raipur
Email: aksharma.maths@nitrr.ac.in

What is R?

- R is a free software environment for statistical computing and graphics
- The R statistical programming language is a free open source package based on the S language developed by Bell Labs.
- The language is very powerful for writing programs.
- Many statistical functions are already built in.
- Large user network contributed packages expand the functionality to cutting edge research.
- Excellent graphics capabilities
- Object-oriented
- Freeware

How to Download R

Firewall Authentication Keepalive | R: The R Project for Statistical Co

r-project.org

[Home]

Download

CRAN

R Project

About R
Logo
Contributors
What's New?
Reporting Bugs
Conferences
Search
Get Involved: Mailing Lists
Developer Pages
R Blog

R Foundation

Foundation
Board
Members
Donors
Donate

Help With R

Getting Help

Documentation

Manuals
FAQs
The R Journal
Books
Certification
Other

Links

Bioconductor
Related Projects
GSoC

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News

- **R version 4.0.2 (Taking Off Again)** has been released on 2020-06-22.
- useR! 2020 in Saint Louis has been cancelled. The European hub planned in Munich will not be an in-person conference. Both organizing committees are working on the best course of action.
- **R version 3.6.3 (Holding the Windsock)** has been released on 2020-02-29.
- You can support the R Foundation with a renewable subscription as a supporting member

News via Twitter

News from the R Foundation

Type here to search

ENG IN 5:09 PM 8/31/2020

Firewall Authentication Keepalive | CRAN - Mirrors

cran.r-project.org/mirrors.html

CRAN Mirrors

The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: main page, windows release, windows old release.

If you want to host a new mirror at your institution, please have a look at the CRAN Mirror HOWTO.

0-Cloud	
https://cloud.r-project.org/	Automatic redirection to servers worldwide, currently sponsored by Rstudio
Algeria	
https://cran.usthb.dz/	University of Science and Technology Houari Boumediene
Argentina	
http://mirror.fcaglp.unlp.edu.ar/CRAN/	Universidad Nacional de La Plata
Australia	
https://cran.csiro.au/	CSIRO
https://mirror.aarnet.edu.au/pub/CRAN/	AARNET
https://cran.ms.unimelb.edu.au/	School of Mathematics and Statistics, University of Melbourne
https://cran.curtin.edu.au/	Curtin University
Austria	
https://cran.wu.ac.at/	Wirtschaftsuniversität Wien
Belgium	
https://www.freestatistics.org/cran/	Patrick Wessa
https://lib.ugent.be/CRAN/	Ghent University Library
Brazil	
https://nbcgib.uesc.br/mirrors/cran/	Computational Biology Center at Universidade Estadual de Santa Cruz
https://cran-r.c3sl.ufpr.br/	Universidade Federal do Parana
https://cran.fiocruz.br/	Oswaldo Cruz Foundation, Rio de Janeiro
https://vps.fmvz.usp.br/CRAN/	University of Sao Paulo, Sao Paulo
https://brieger.esalq.usp.br/CRAN/	University of Sao Paulo, Piracicaba
Bulgaria	
https://ftp.uni-sofia.bg/CRAN/	Sofia University
Canada	
https://mirror.rcg.sfu.ca/mirror/CRAN/	Simon Fraser University, Burnaby
https://muug.ca/mirror/cran/	Manitoba Unix User Group
https://mirror.its.dal.ca/cran/	Dalhousie University, Halifax
http://cran.utstat.utoronto.ca/	University of Toronto
Chile	
https://cran.dcc.uchile.cl/	Departamento de Ciencias de la Computación, Universidad de Chile
https://cran.dme.ufro.cl/	Departamento de Matemática y Estadística, Universidad de La Frontera
China	
https://mirrors.tuna.tsinghua.edu.cn/CRAN/	TUNA Team, Tsinghua University
https://mirrors.bfsu.edu.cn/CRAN/	Beijing Foreign Studies University
https://mirrors.ustc.edu.cn/CRAN/	University of Science and Technology of China
https://mirror-hk.koddos.net/CRAN/	KoDDoS in Hong Kong
https://mirrors.e-education.cn/CRAN/	Elite Education
https://mirror.lzu.edu.cn/CRAN/	Lanzhou University Open Source Society
https://mirror.nju.edu.cn/CRAN/	eScience Center, Nanjing University

Type here to search

ENG IN 5:10 PM 8/31/2020

Firewall Authentication Keepalive | The Comprehensive R Archive Ne

cloud.r-project.org

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

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
- Download R for (Mac) OS X
- 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 (2020-06-22, Taking Off Again) R-4.0.2.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- 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 terms are, please read our answers to frequently asked questions before you send an email.

What are R and CRAN?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the R project homepage for further information.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN mirror nearest to you to minimize network load.

Submitting to CRAN

To "submit" a package to CRAN, check that your submission meets the CRAN Repository Policy and then use the web form.

If this fails, upload to ftp://CRAN.R-project.org/incoming/ and send an email to CRAN-submissions@R-project.org following the policy. Please do not attach submissions to emails, because this will clutter up the mailboxes of half a dozen people.

Note that we generally do not accept submissions of precompiled binaries due to security reasons. All binary distribution listed above are compiled by selected maintainers, who are in charge for all binaries of their platform, respectively.

This year's winter break will be from Dec 18, 2020 to Jan 4, 2021.

For queries about this web site, please contact the webmaster.

Type here to search

ENG IN 5:11 PM 8/31/2020

Firewall Authentication Keepalive | The Comprehensive R Archive Ne

cloud.r-project.org

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

R for Windows

Subdirectories:

base	Binaries for base distribution. This is what you want to install R for the first time.
contrib	Binaries of contributed CRAN packages (for R >= 2.13.x; managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).
Rtools	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the R FAQ and R for Windows FAQ.

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

Type here to search

ENG IN 5:15 PM 8/31/2020

Firewall Authentication Keepalive | The Comprehensive R Archive Ne

cloud.r-project.org

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

R-4.0.2 for Windows (32/64 bit)

Download R 4.0.2 for Windows (84 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server. You will need a version of md5sum for windows: both graphical and command line versions are available.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Other builds

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is <CRAN MIRROR>/bin/windows/base/release.html.

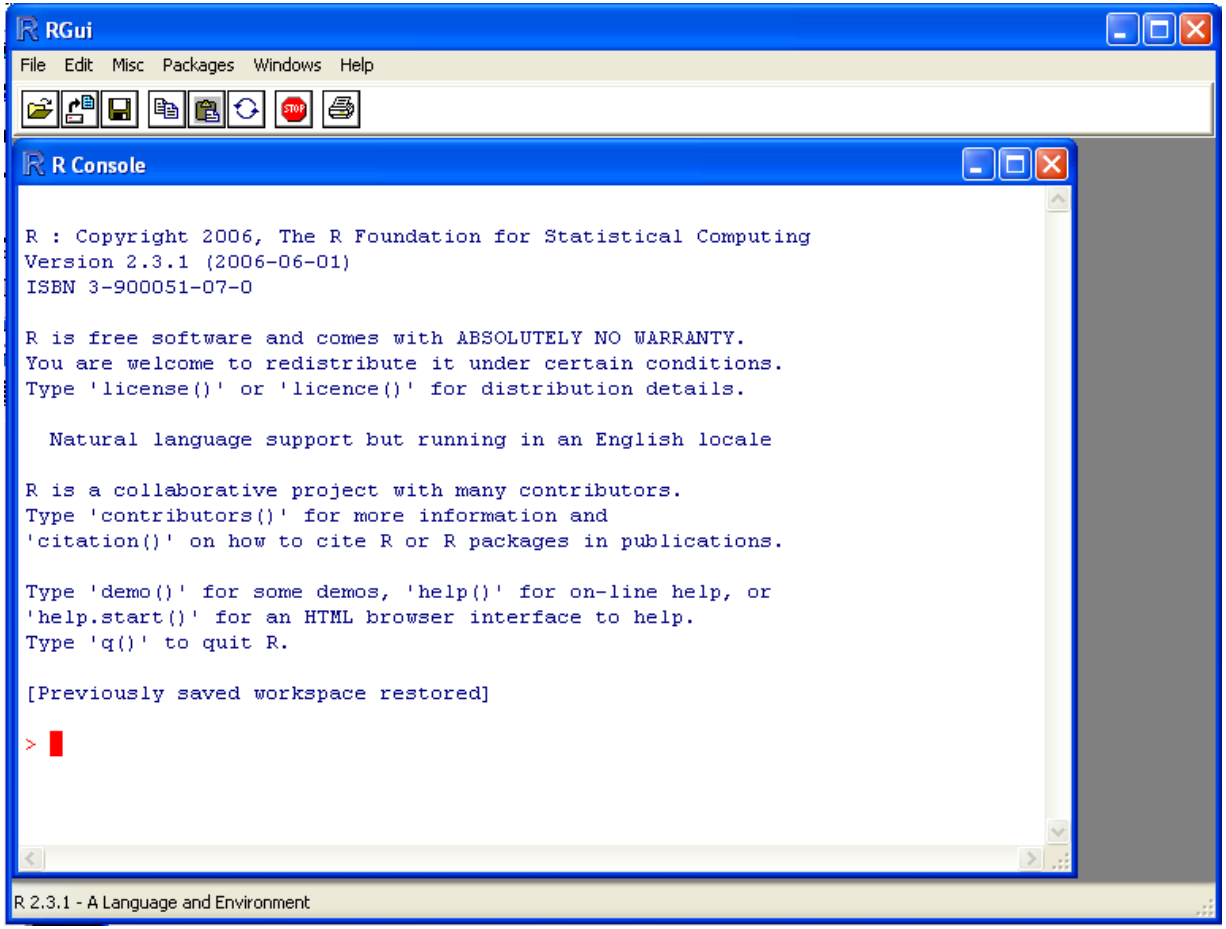
Last change: 2020-06-22

Type here to search

ENG IN 5:17 PM 8/31/2020

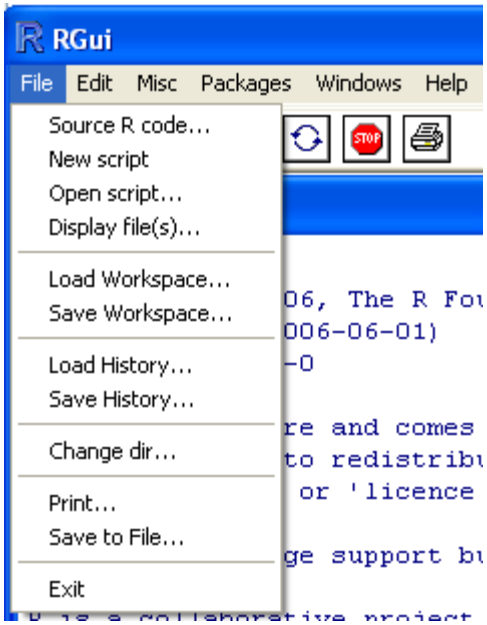
Getting Started

- The R GUI?



Getting Started

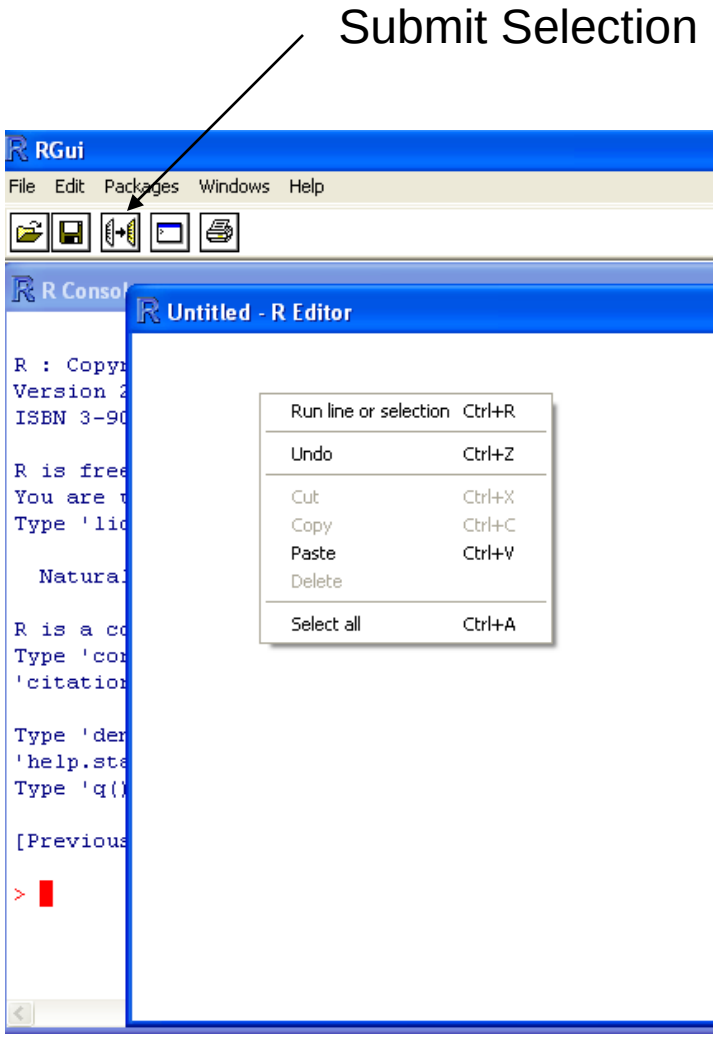
- Opening a script.
- This gives you a script window.



Getting Started

- Submitting a program:
- Use button

- Right mouse click and run selection.



RStudio

What is RStudio?

- RStudio is an open-source Integrated Development Environment (IDE) that facilitates.

Code-Colouring, Code-Complition and Debugging

- There are two versions of RStudio – **RStudio Desktop** and **RStudio Server**.
- RStudio desktop provides facilities for working on the local desktop environment, whereas RStudio Server provides access through a web browser.

RStudio

File Edit Code View Plots Session Project Build Tools Help

Console H:/MyData/RFiles/

```
R version 3.0.0 (2013-04-03) -- "Masked Marvel"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

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.

> getwd()
[1] "H:/MyData/RFiles"
> 5*5
[1] 25
> A <- matrix(c(1,2,3,4,5,6,7,8), nrow=4, ncol=2)
> A
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
> B <- matrix(c(1,2,3,4,5,6,7,8), nrow=4, ncol=2, byrow=TRUE)
> B
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
>
```

The **console** is where you can type commands and see output

Workspace History

Import Dataset

Data	
A	4x2 double matrix
B	4x2 double matrix

The **workspace** tab shows all the active objects (see next slide). The **history** tab shows a list of commands used so far.

Files Plots Packages Help

New Folder Delete Rename More

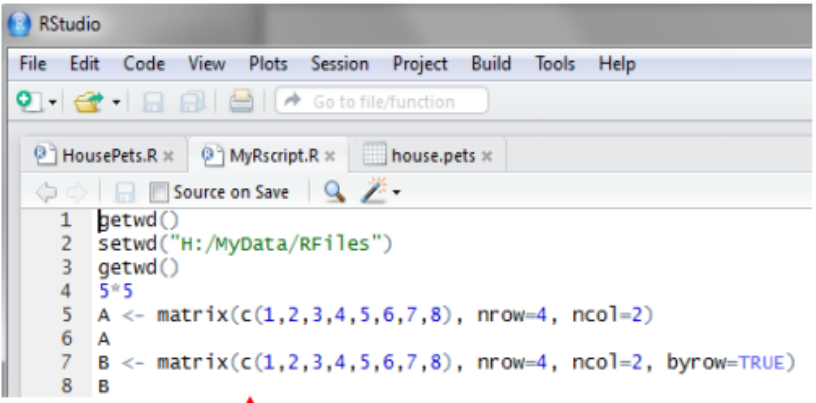
H: MyData RFiles

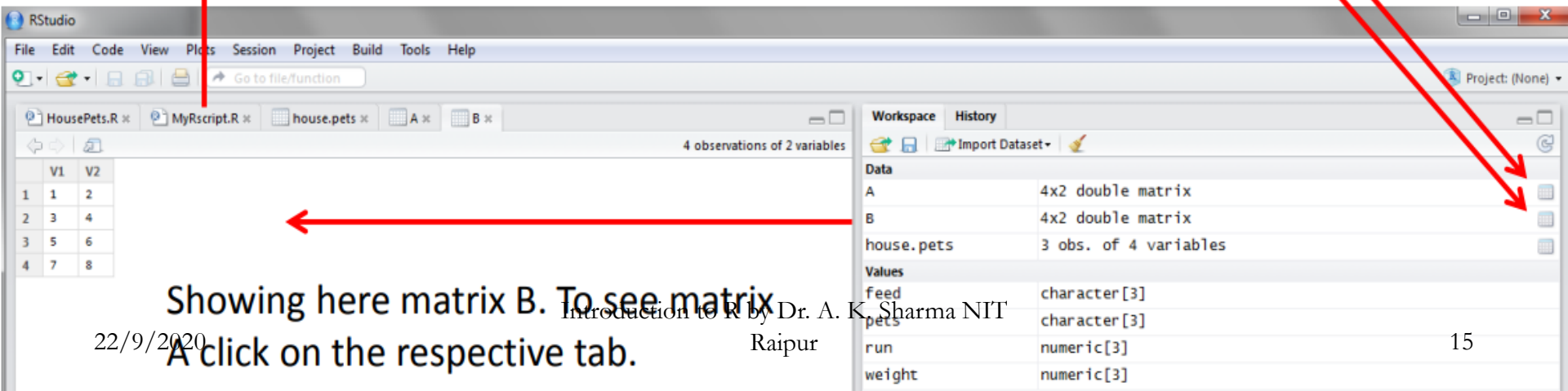
Name	Size	Modified
..		
.Rhistory	34 bytes	Aug 23, 2013, 1:26 PM

The **files** tab shows all the files and folders in your default workspace as if you were on a PC/Mac window. The **plots** tab will show all your graphs. The **packages** tab will list a series of packages or add-ons needed to run certain processes. For additional info see the **help** tab

Workplace

The workspace tab stores any object, value, function or anything you create during your R session. In the example below, if you click on the dotted squares you can see the data on a screen to the left.





Showing here matrix B. To see matrix A click on the respective tab.

R Operators

- Assignment operators
- Arithmetic operators
- Comparison operators
- Logical operators
- Element-wise Logical operators
- Membership operators

Assignment Operators

Operator	Meaning	Example
=	Assignment	x = 3
<-, <<-, =	Leftwards assignment	x <- 3, x <<- 3, x = 3
->, ->>	Rightwards assignment	3 -> x, 3 ->> x

Arithmetic Operators

Operator	Meaning	Example
+	Addition	x + y
–	Subtraction	x – y
*	Multiplication	x * y
/	Division	x / y
%%	Modulus	x %% y
^	Exponents	x ^ y
%/%	Integer division	x %/% y

Comparison Operators

Operator	Meaning	Example
==	Equal to	x == y
!=	Not equal to	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	Example
&&	Returns True if both statements are true	x > 0 && y < 0
||	Returns True if one of the statements is true	x > 0 || y < 0
!	Reverses the result, returns False if the result is true	!(x > 0 && y < 0)

Element-wise Logical Operators

Operator	Description	Example
&	Returns True if respective elements of both vectors are true	v1 && v2
|	Returns True if one of the respective elements of both vectors is true	v1 || v2

Membership Operator

Operator	Description	Example
%in%	Returns True if a value is present in the vector or the list	x %in% y

Miscellaneous Operators

Operator	Description	Example
:	Generates a number sequence from a to b	1:10
%*%	Multiplies two matrices	m1 %*% m2

Operator Precedence (Order of Operations)

	Operator	Description
highest precedence	({	Function calls and grouping expressions (respectively)
	[[[	Indexing
	:: :::	Access variables in a namespace
	$ @	Component / slot extraction
	^	Exponentiation (right to left)
	– +	Unary minus and plus
	:	Sequence operator
	%any%	Special operators
	* /	Multiply, divide
	+ –	(Binary) add, subtract
	< > <= >= == !=	Ordering and comparison
	!	Negation
	& &&	And
	| ||	Or
	~	As in formulas
	-> ->>	Rightward assignment
	=	Assignment (right to left)
	<- <<-	Assignment (right to left)
lowest precedence	?	Help (unary and binary)

R Vector

A vector is a collection of same type elements.

Types of Vectors:

- ❑ Logical
- ❑ Character
- ❑ Integer and
- ❑ Double (or numeric).

Create a vector: **c()**

Create a Sequence: **“ : “ , seq(), rep()**

Change the Vector Type: **as.vector()**

Naming a Vector: **names()**

R Matrix

A matrix is a collection of elements, all the same type, arranged in a two-dimensional layout.

In a nutshell, a matrix is just a vector that has two dimensions.

When using R, you will frequently encounter the four basic matrix types viz. logical, character, integer and double (often called numeric).

Create a Matrix: **matrix(),**
matrix(argument, nrow=m, ncol=n),
matrix(argument, nrow=m, ncol=n, byrow=TRUE)

R List

Vectors and matrices are incredibly useful data structure in R, but they have one distinct limitation: they can store only one type of data.

Lists, however, can store multiple types of values at once. A list can contain a numeric matrix, a logical vector, a character string, a factor object and even another list.

Create a list: **list()**

Creating a list is much like creating a vector; just pass a comma-separated sequence of elements to the list() function.

```
lst <- list(1, 2, 3)
lst <- list("red", "green", "blue")
lst <- list(1, "abc", 1.23, TRUE)
```

R Data Frame

That data structure is a Data Frame.

Unlike vectors or matrices, data frames have no restriction on the data types of the variables; you can store numeric data, character data, and so on.

In a nutshell, a data frame is a list of equal-length vectors.

Data frame example: Excel worksheet.

Create Data Frame: **data.frame()**

```
# Create a data frame to store employee records

name <- c("Bob", "Max", "Sam")
age <- c(25,26,23)
city <- c("New York", "Chicago", "Seattle")

df <- data.frame(name, age, city)
df
```

```
  name  age     city
1  Bob   25 New York
2  Max   26  Chicago
3  Sam   23  Seattle
```

R if else Statement

Often, we need to execute some statements only when some condition is met. You can use following conditional statements in your code to do this.

if Statement: use it to execute a block of code, if a specified condition is true
else Statement: use it to execute a block of code, if the same condition is false
else if Statement: use it to specify a new condition to test, if the first condition is false
If else() Function: use it when to check the condition for every element of a vector

Condition

Any expression that evaluates to true or false

```
if (condition) {
    statement
    statement
      ...
}
following_statement
```

True branch

This is executed if the condition is true

```
x <- 7
y <- 5
if(x > y) {
  print("x is greater")
}
```

```
[1] "x is greater"
```

```
if (condition) {
    statement
    statement
      ...
} else {
    statement
    statement
      ...
}
following_statement
```

True branch

This is executed if the condition is true

False branch

This is executed if the condition is false

```
x <- 7
y <- 5
if(x > y) {
  print("x is greater")
} else {
  print("y is greater")
}
```

```
[1] "x is greater"
```

```
if (condition) {
    statement
    statement
        ...
} else if (condition) {
    statement
    statement
        ...
} else {
    statement
    statement
        ...
}
following_statement
```

First condition

This is executed if the first condition is true

New condition

A new condition to test if previous condition isn't true

False branch

This is executed if none of the conditions are true

```
x <- 5
y <- 5
if(x > y) {
  print("x is greater")
} else if(x < y) {
  print("y is greater")
} else {
  print("x and y are equal")
}
```

```
[1]  "x and y are equal"
```

```
ifelse (condition, TrueVector, FalseVector)
```

Condition	True branch	False branch
Condition is checked for every element of a vector	Select element from this if the condition is true	Select element from this if the condition is false

```
v <- c(1,2,3,4,5,6)
ifelse(v %% 2 == 0, "even", "odd")
```

```
[1] "odd"  "even" "odd"  "even" "odd"  "even"
```

R For Loop

Var

It takes items from iterable one by one

Iterable

It's a collection of objects (like a vector, list etc.)

```
for (var in iterable) {
    statement
    statement
        ...
}
following_statement
```

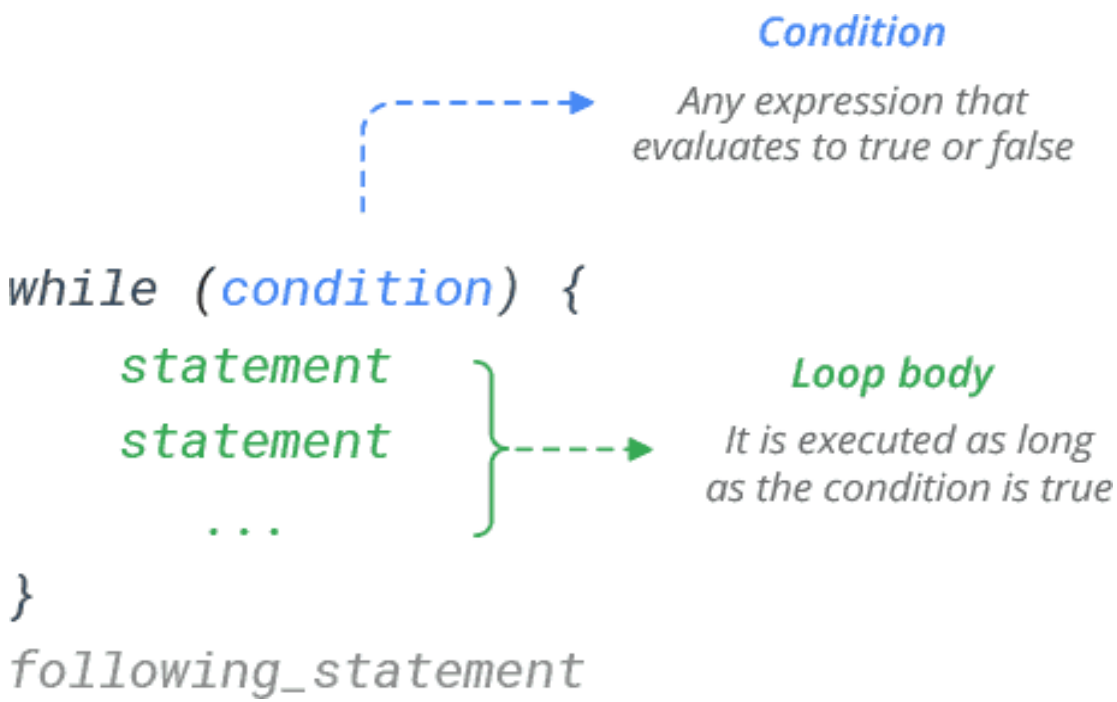
Loop body

It is executed once for each item in iterable

```
# Print 'Hello!' 3 times

for (x in 1:3) {
  print("Hello!")
}

[1] "Hello!"
[1] "Hello!"

# Iterate through a vector

colors <- c("red","green","blue","yellow")
for (x in colors) {
  print(x)
}

[1] "red"
[1] "green"
[1] "blue"
[1] "yellow"
```

R While Loop

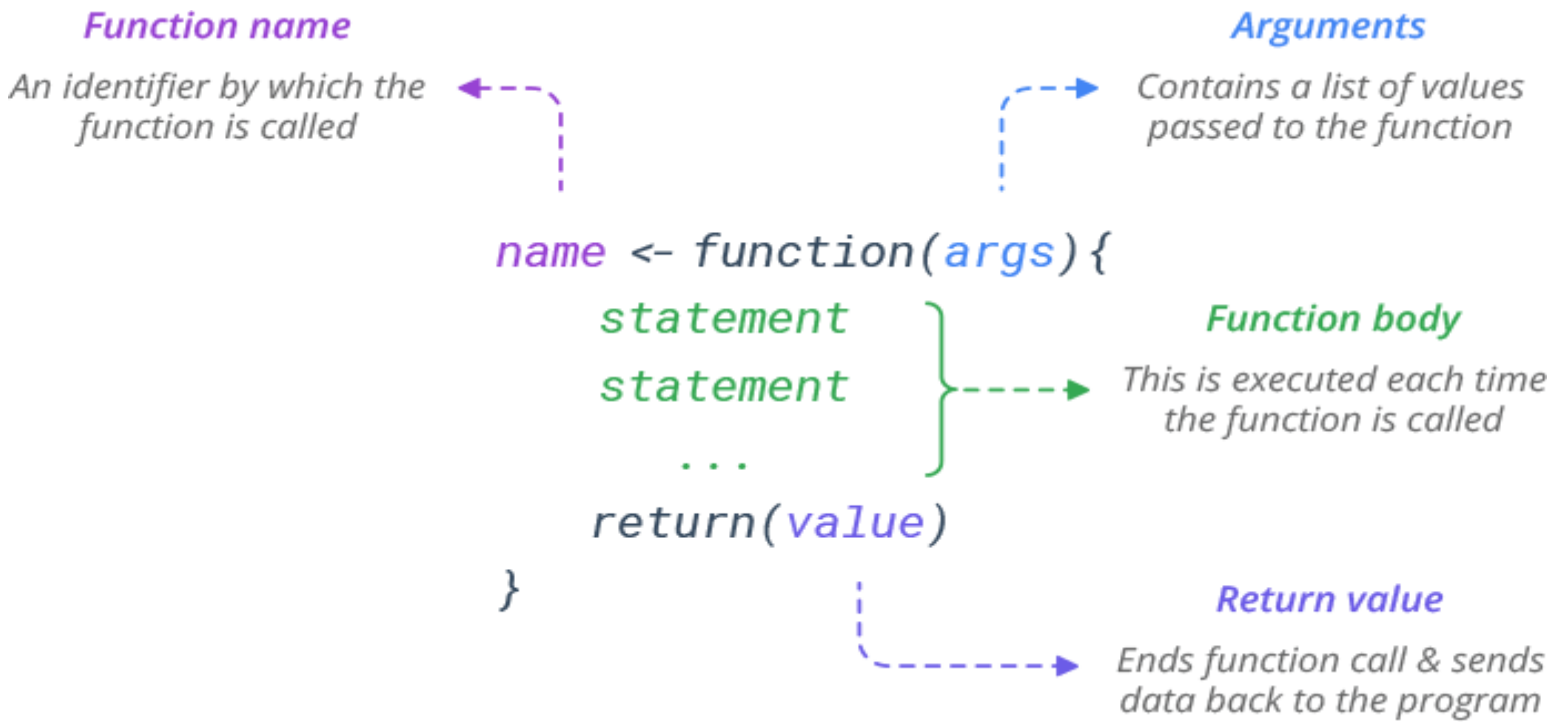
A while loop is used when you want to perform a task indefinitely, until a particular condition is met. It's a condition-controlled loop.



```
x <- 5
while (x>0) {
      print(x)
      x <- x - 1
}
1] 1
[1] 2
[1] 3
[1] 4
[1] 5

# Skip odd numbers using continue statement
x <- 6
while (x) {
  x <- x - 1
  if (x %% 2 != 0)
    next
  print(x)
}

[1] 4
[1] 2
[1] 0
```

R Functions

A function is a block of statements that can be used repeatedly in a program. R provides many built-in functions and allows programmers to define their own functions.



```
sum <- function(x, y) {
  x + y
}

sum(2, 3)
[1] 5


math <- function(x, y)
 { add <- x + y
sub <- x - y
mul <- x * y div <- x / y
c(addition = add, subtraction = sub, multiplication = mul, division = div)
 }
math(6, 3)
addition    subtraction   multiplication    division
9                3                 18                2
```

Read and Write Excel Files in R

Excel is the most popular spreadsheet software used to store tabular data. So, it's important to be able to efficiently import and export data from these files.

R's xlsx package makes it easy to read, write, and format excel files.

```
# Install and load xlsx package
install.packages("xlsx")
library("xlsx")
```

You can read the contents of an Excel worksheet using the **read.xlsx()** or **read.xlsx2()** function.

The **read.xlsx()** function reads the data and creates a data frame.

Both the functions work exactly the same except, **read.xlsx()** is slow for large data sets (worksheet with more than 100 000 cells).
On the contrary, **read.xlsx2()** is faster on big files.

Read an Excel file

	A	B	C	D	E
1	name	age	job	city	
2	Bob	25	Manager	Seattle	
3	Sam	30	Developer	New York	
4	Amy	20	Developer	Houston	
5					
6					
7					

```
library(xlsx)
mydata <- read.xlsx("mydata.xlsx", header = TRUE)
mydata <- read.xlsx("mydata.xlsx", sheetIndex=1)
mydata
```

```
name age      job     city
1  Bob  25   Manager  Seattle
2  Sam  30 Developer New York
3  Amy  20 Developer  Houston
```

Write Data to an Excel File

To write to an existing file, use **write.xlsx()** method and pass the data in the form of matrix or data frame.

```
# Export data from R to an excel workbook
df
  name age       job     city
1  Bob  25   Manager  Seattle
2  Sam  30 Developer New York
3  Amy  20 Developer  Houston

write.xlsx(df, file = "mydata.xlsx")
```

	A	B	C	D	E	F
1		name	age	job	city	
2	1	Bob	25	Manager	Seattle	
3	2	Sam	30	Developer	New York	
4	3	Amy	20	Developer	Houston	
5						
6						
7						

Read and Write CSV Files in R

The CSV file (Comma Separated Values file) is a widely supported file format used to store tabular data.

```
mydata <- read.csv("mydata.csv" , header = TRUE)
mydata
  name age       job     city
1  Bob  25   Manager  Seattle
2  Sam  30 Developer New York
```

Remember! while specifying the exact path, characters prefaced by \ (like \n \r \t etc.) are interpreted as special characters.

You can escape them by:

Changing the backslashes to forward slashes like: "C:/data/myfile.csv"
Using the double backslashes like: "C:\\data\\myfile.csv"

Write a CSV File

To write to an existing file, use **write.csv()** method and pass the data in the form of matrix or data frame.

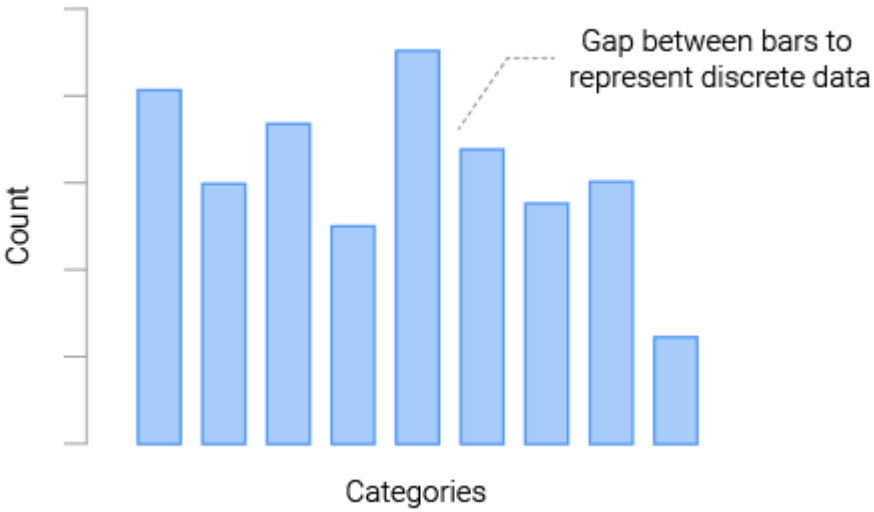
```
# Write a CSV File from a data frame
df
  name age       job     city
1  Bob  25   Manager  Seattle
2  Sam  30 Developer New York

write.csv(df, "mydata.csv")
```

mydata.csv

```
"","name","age","job","city"
"1","Bob","25","Manager","Seattle"
"2","Sam","30","Developer","New York"
```

R Bar Plot – Base Graph

A Bar Graph (or a Bar Chart) is a graphical display of data using bars of different heights.

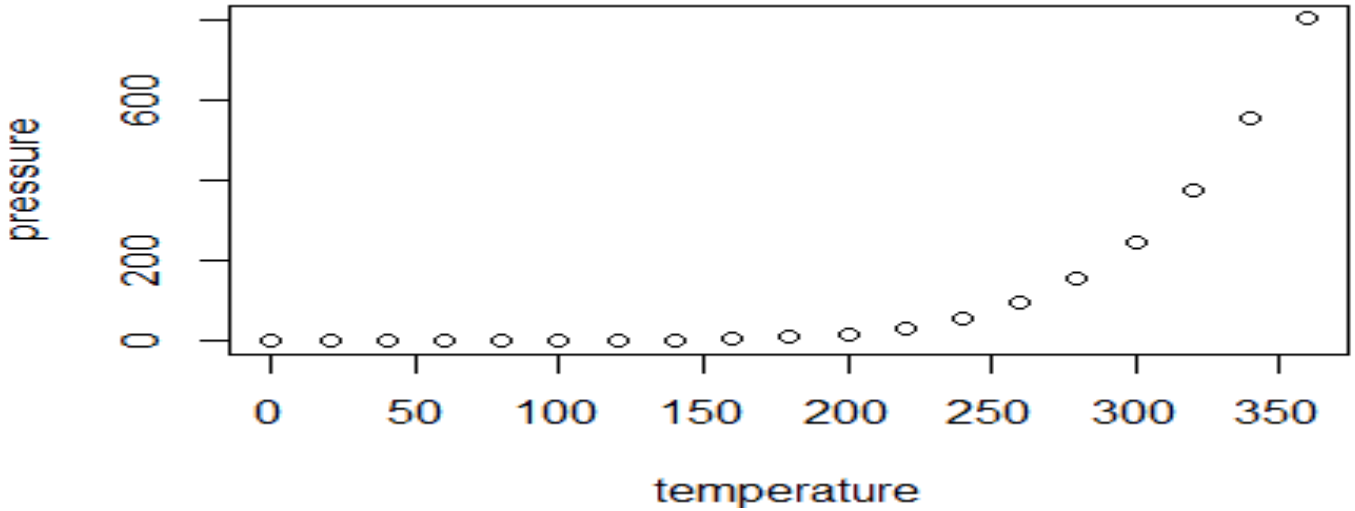


barplot(x,y,type,main,xlab,ylab,pch,col,las,bty,bg,cex,…)

Parameter	Description
x	The coordinates of points in the plot
y	The y coordinates of points in the plot
type	The type of plot to be drawn
main	An overall title for the plot
xlab	The label for the x axis
ylab	The label for the y axis
pch	The shape of points
col	The foreground color of symbols as well as lines
las	The axes label style
bty	The type of box round the plot area
bg	The background color of symbols (only 21 through 25)
cex	The amount of scaling plotting text and symbols
…	Other graphical parameters

```
# First six observations of the 'Pressure' dataset
head(pressure)
  temperature pressure
1           0   0.0002
2          20   0.0012
3          40   0.0060
4          60   0.0300
5          80   0.0900
6         100   0.2700
```

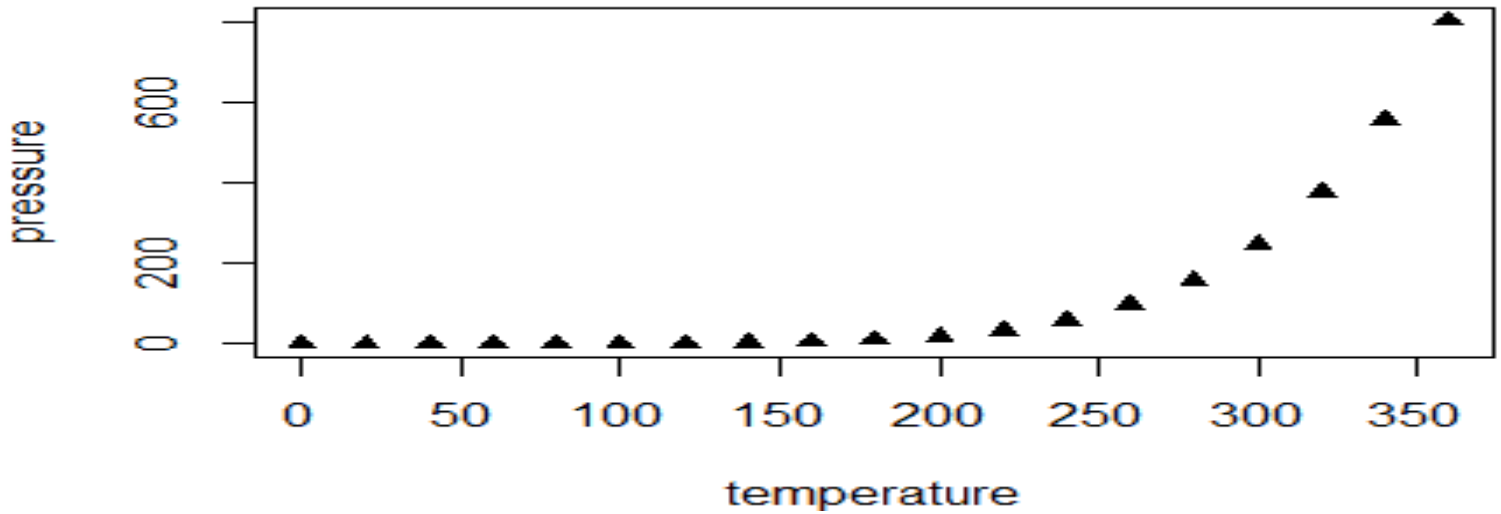
```
# Plot the 'pressure' dataset
plot(pressure)
```



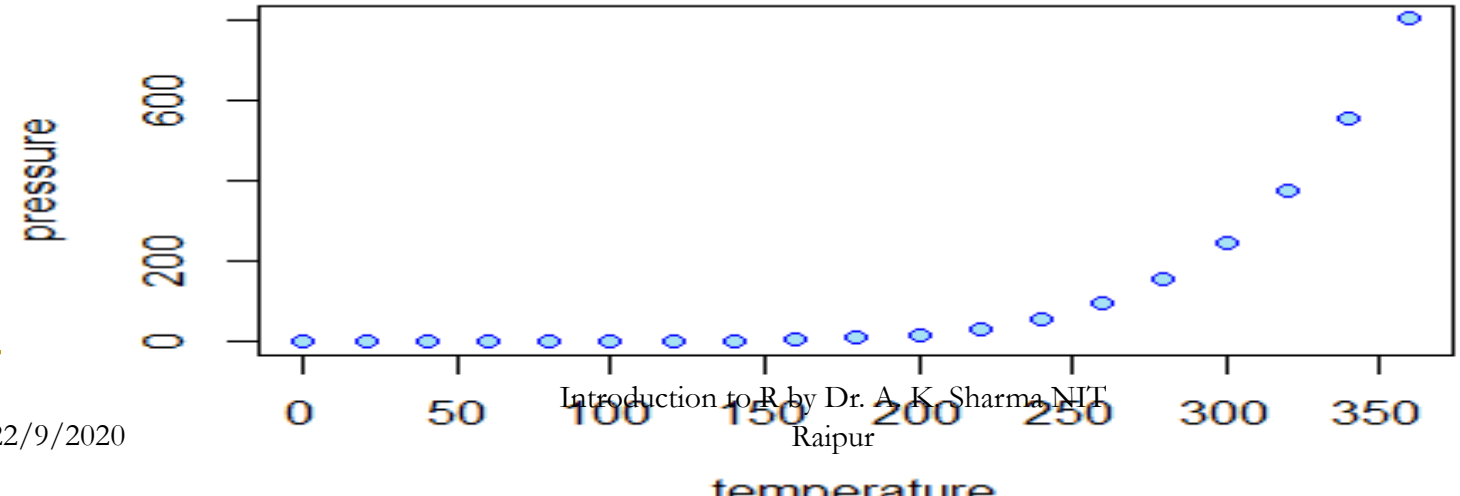
Change the Shape and Size of the Points

You can use the pch (plotting character) argument to specify symbols to use when plotting points.

# Change the shape of the points
plot(pressure, pch=17)



# Change the border color to blue and background color to light blue
plot(pressure, pch=21, col="blue", bg="lightblue")



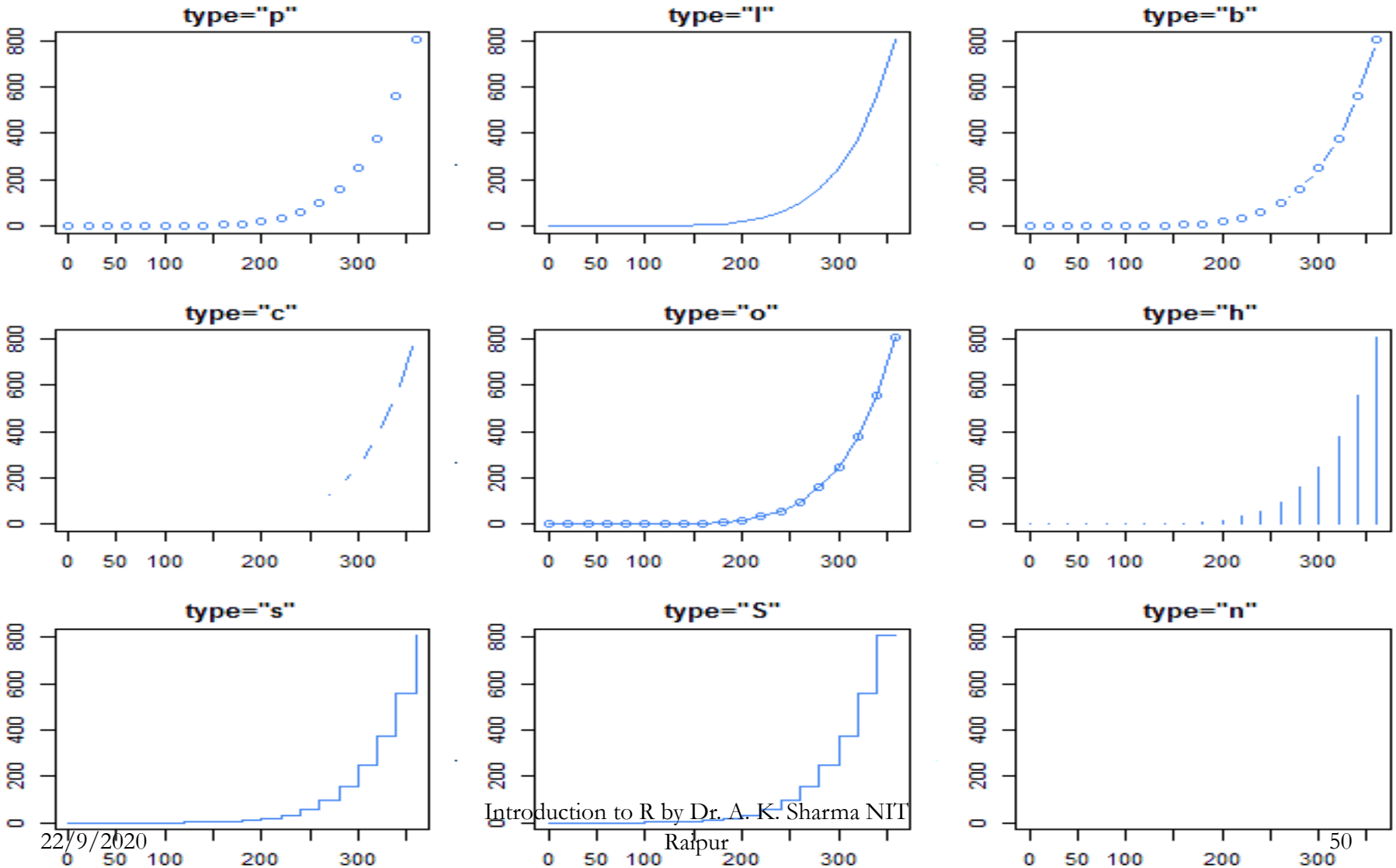
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125
126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225
226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250
251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275
276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325
326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350
351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375
376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400
401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425
426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450
451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475
476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500
501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525
526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550
551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575
576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600
601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625
626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650
651	652	653	654	655	656	657																		

Different Plot Types

You can change the type of plot that gets drawn by using the type argument.

Value	Description
“p”	Points
“l”	Lines
“b”	Both points and lines
“c”	The lines part alone of “b”
“o”	Both points and lines “overplotted”
“h”	Histogram like (or high-density) vertical lines
“s”	Step plot (horizontal first)
“S”	Step plot (vertical first)
“n”	No plotting

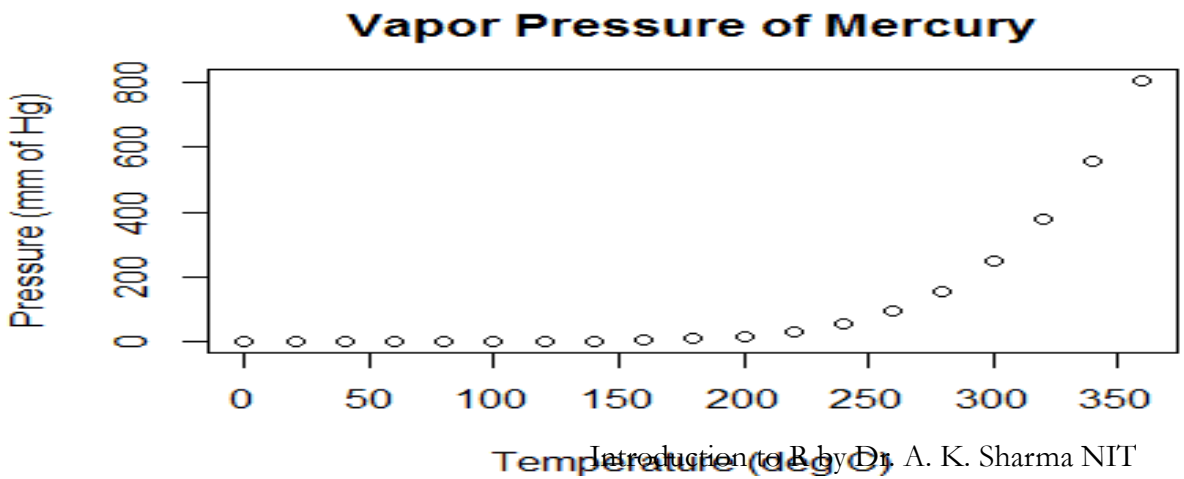
For example, to create a plot with lines between data points, use type="l"; to draw both lines and points, use type="b".
A series of graphics showing different types is shown below.



Adding Titles and Axis Labels

Argument	Description
main	Main plot title
xlab	x-axis label
ylab	y-axis label

```
plot(pressure,
     main = "Vapor Pressure of Mercury",
     xlab = "Temperature (deg C)",
     ylab = "Pressure (mm of Hg)")
```

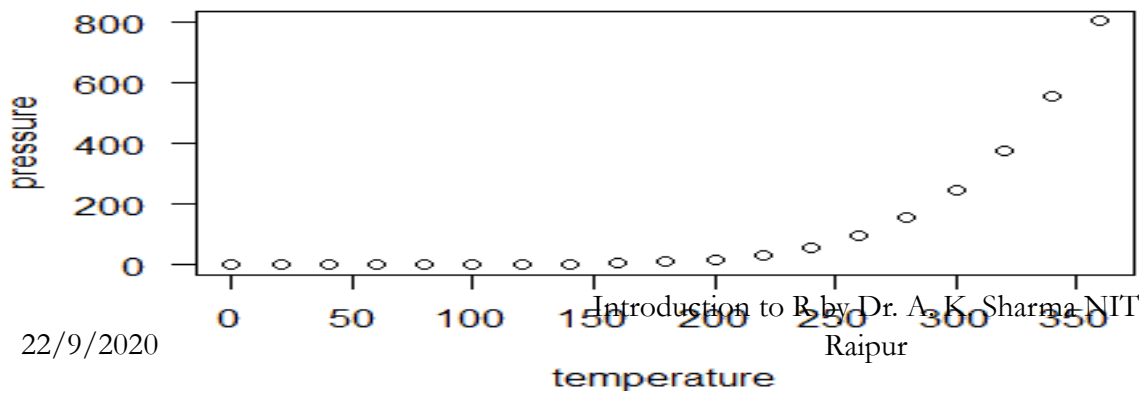


The Axes Label Style

By specifying the las (label style) argument, you can change the axes label style. This changes the orientation angle of the labels.

Value	Description
0	The default, parallel to the axis
1	Always horizontal
2	Perpendicular to the axis
3	Always vertical

plot(pressure, las = 1)

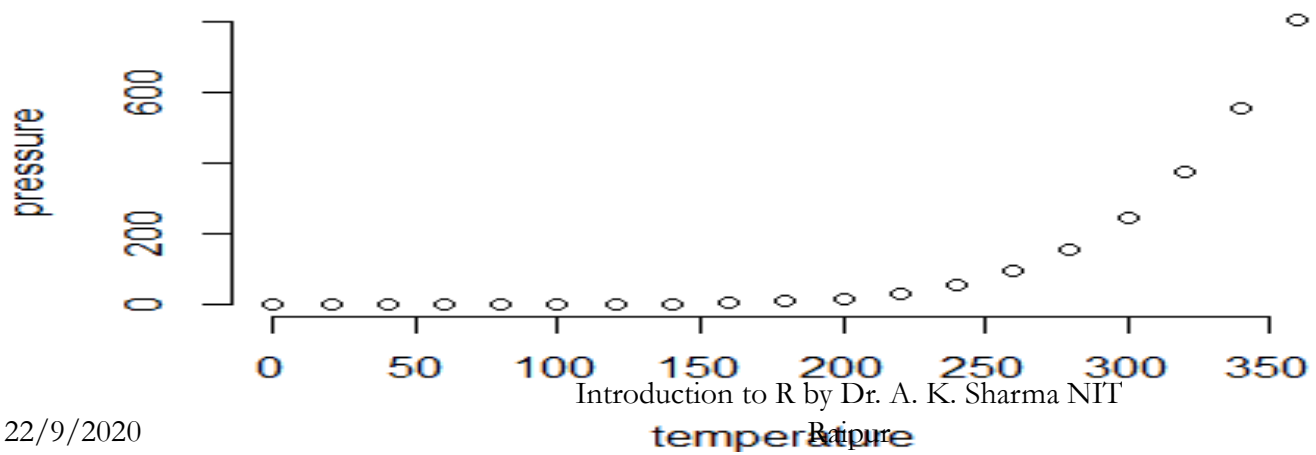


The Box Type

Specify the bty (box type) argument to change the type of box round the plot area.

Value	Description
"o"	(default) Draws a complete rectangle around the plot.
"n"	Draws nothing around the plot.
"l", "7", "c", "u", or "]"	Draws a shape around the plot area.

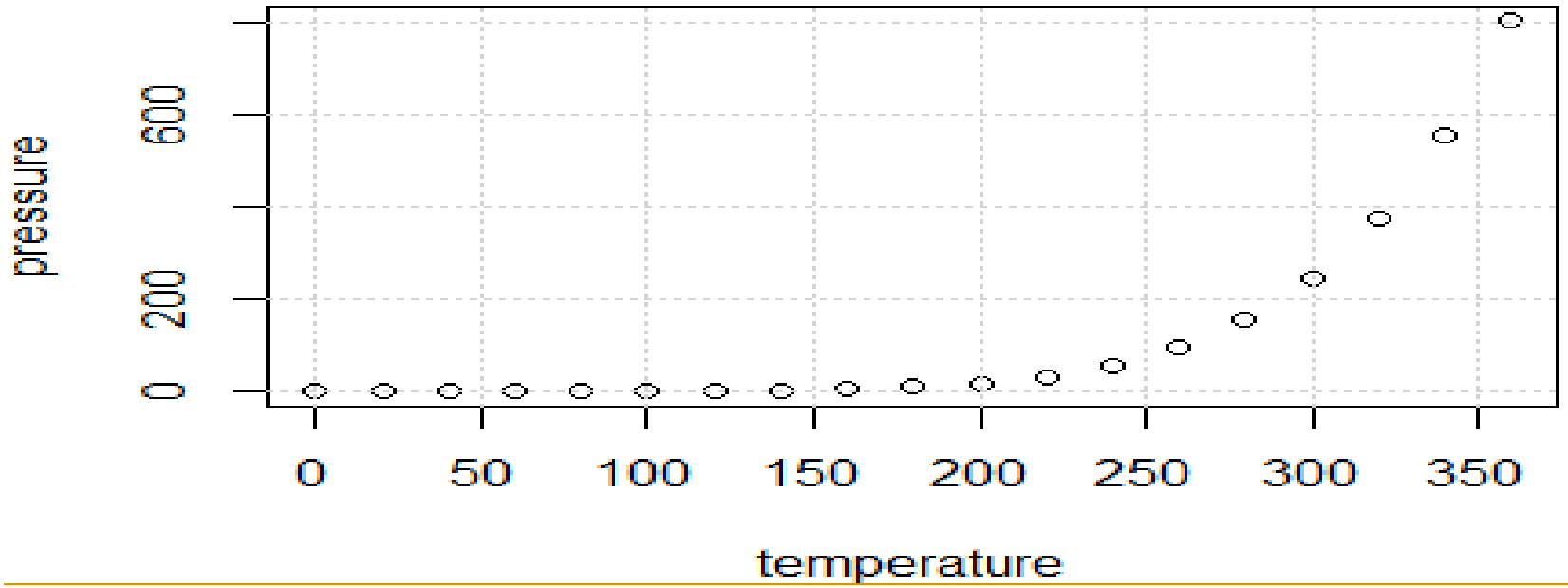
```
# Remove the box round the plot
plot(pressure, bty="n")
```



Add a Grid

The plot() function does not automatically draw a grid. However, it is helpful to the viewer for some plots. Call the grid() function to draw the grid once you call the plot().

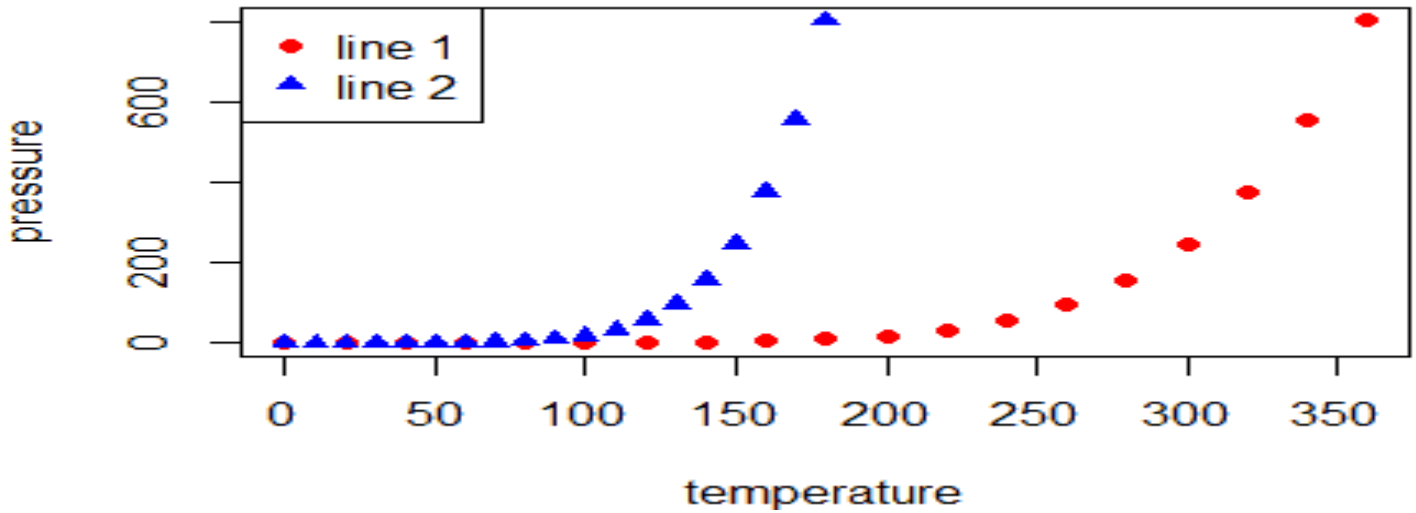
```
plot(pressure)
grid()
```



Add a Legend

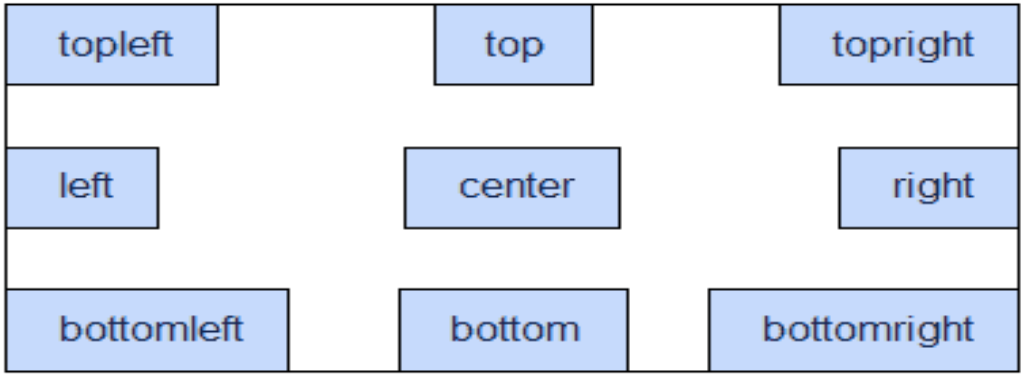
You can include a legend to your plot – a little box that decodes the graphic for the viewer. Call the legend() function, once you call the plot().

```
# Add a legend to the top left corner
plot(pressure, col="red", pch=19)
points(pressure$temperature/2, pressure$pressure,col="blue", pch=17)
legend("topleft", c("line 1","line 2"), pch=c(19,17), col=c("red","blue"))
```



The position of the legend can be specified using the following keywords : “bottomright”, “bottom”, “bottomleft”, “left”, “topleft”, “top”, “topright”, “right” and “center”.

The effect of using each of these keywords is shown below.

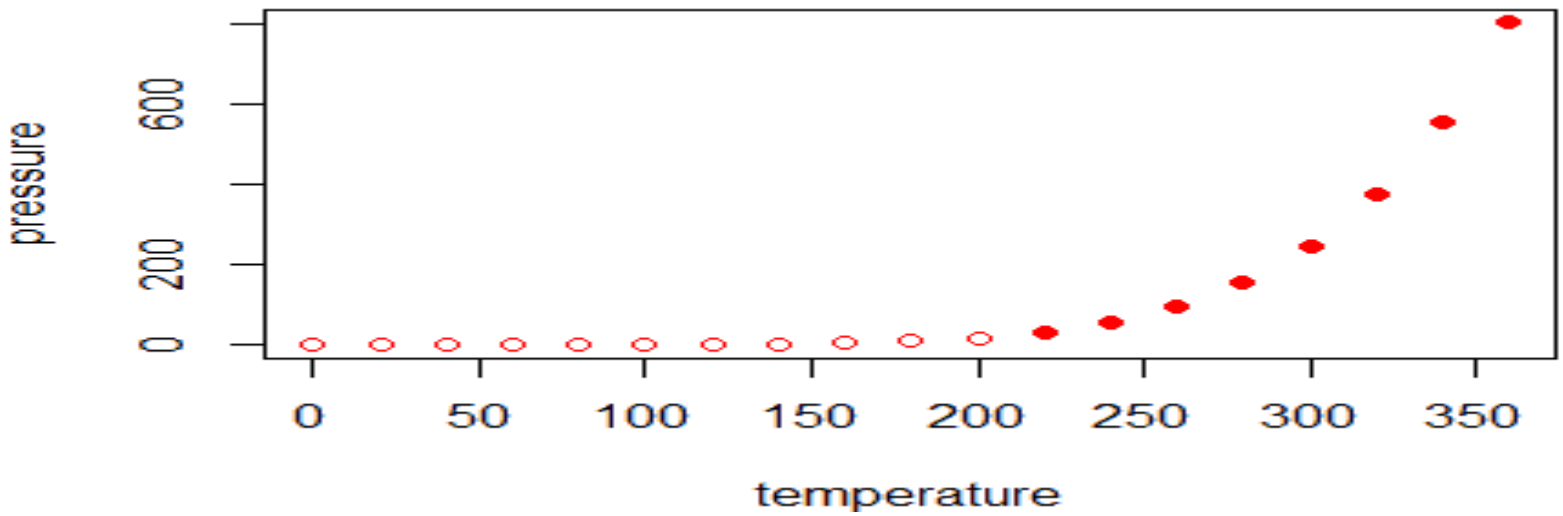


Add Points to a Plot

You can add points to a plot with the points() function.

For example, let's create a subset of pressure containing temperatures greater than 200 °C and add these points to the plot.

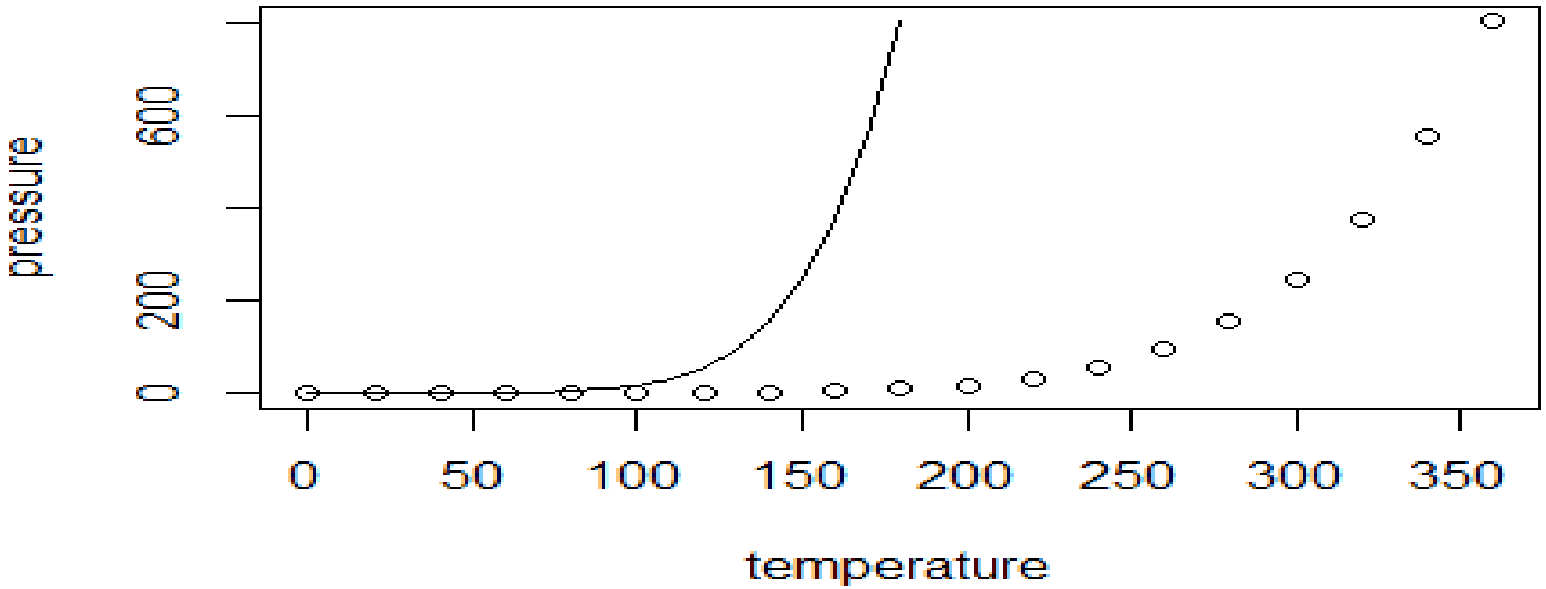
```
plot(pressure, col = "red")
points(pressure[pressure$temperature > 200, ], col = "red", pch = 19)
```



Add Lines to a Plot

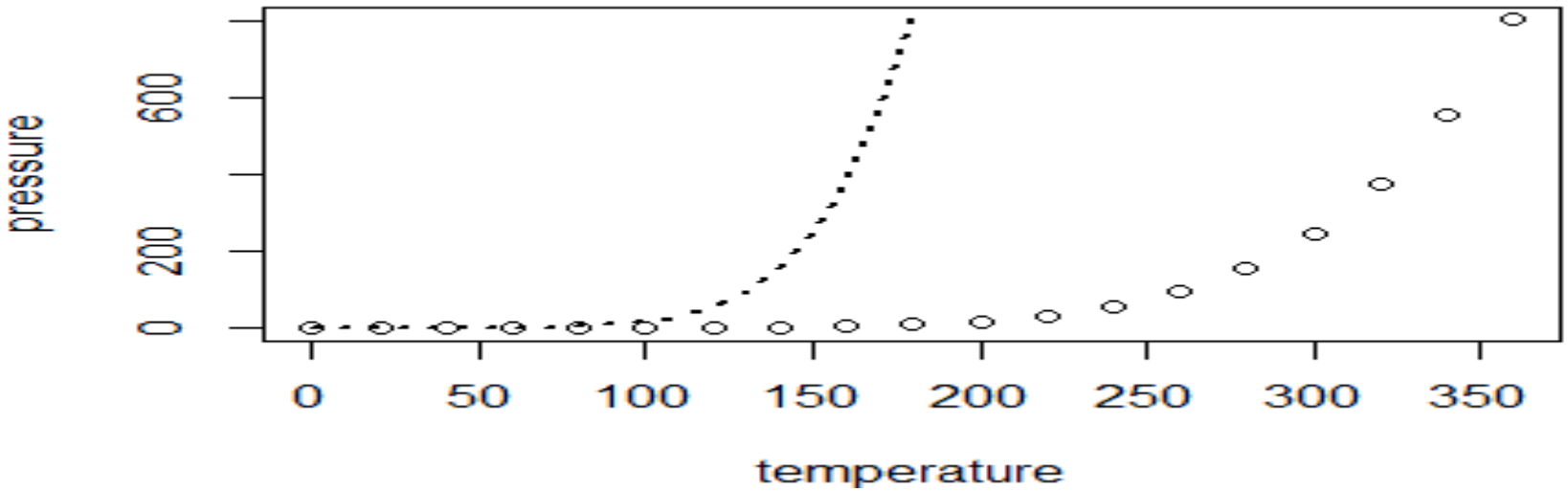
You can add lines to a plot in a very similar way to adding points, except that you use the lines() function to achieve this.

```
plot(pressure)
lines(pressure$temperature/2, pressure$pressure)
```



You can change the line type using lty argument; and the line width using lwd argument.

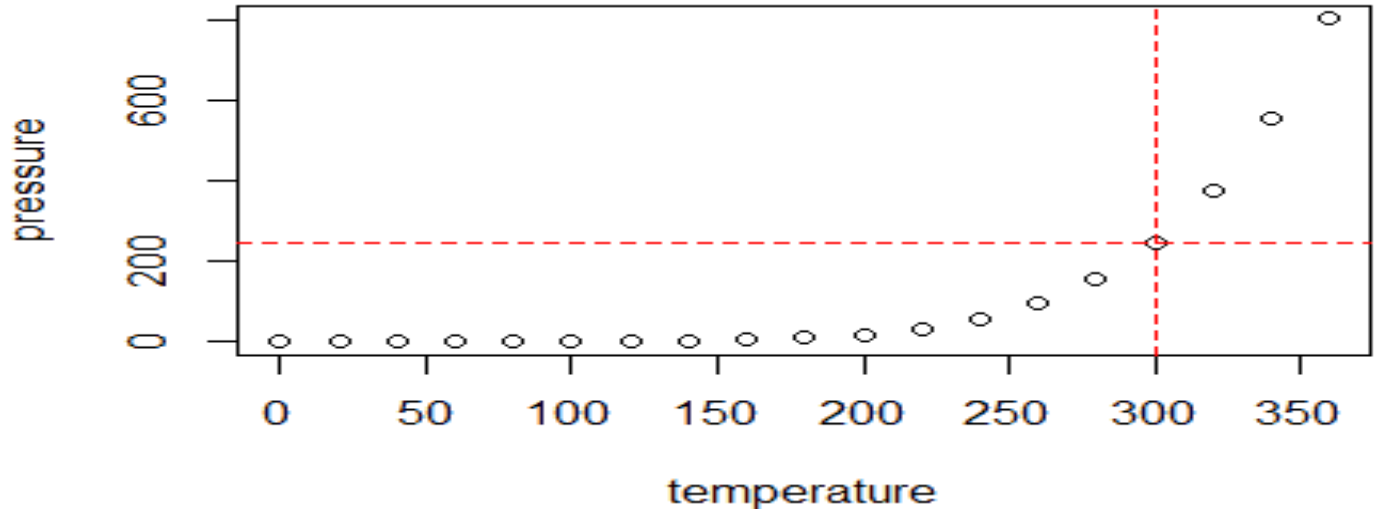
```
# Change the line type and line width
plot(pressure)
lines(pressure$temperature/2, pressure$pressure, lwd=2, lty=3)
```



Here's a list of line types you can use.

There's another function called abline() which allows you to draw horizontal, vertical, or sloped lines.

```
# Draw a dotted horizontal line at 247 and vertical line at 300
plot(pressure)
abline(h= 247, v=300, col="red", lty=2)
```

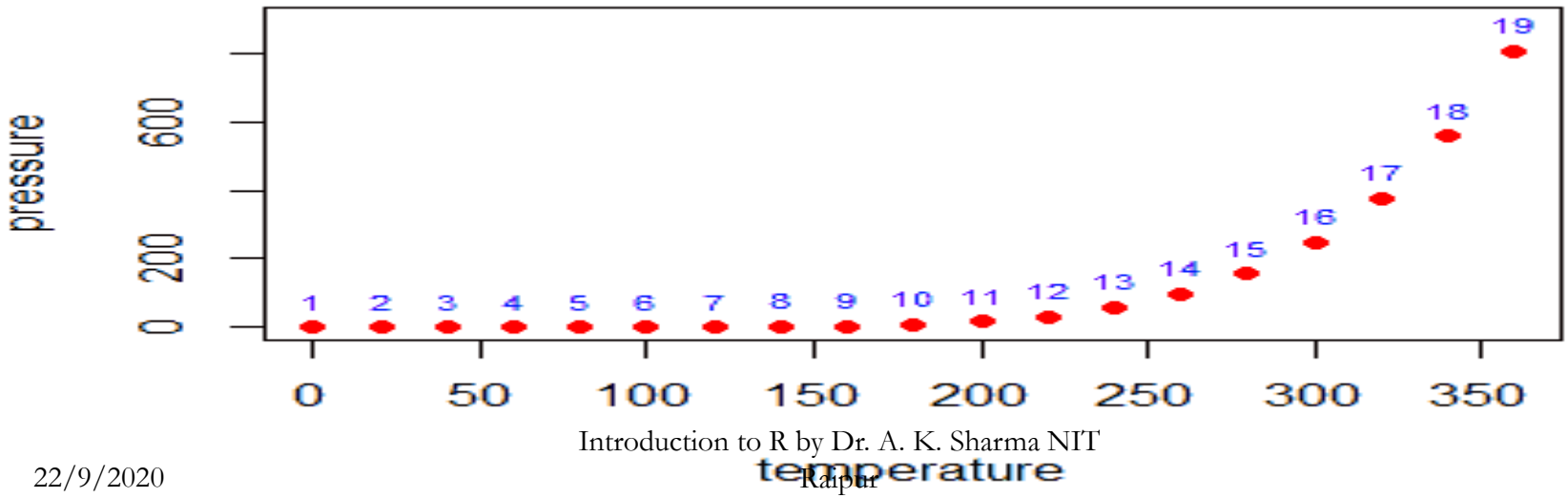

pressure
0
200
600
0
50
100
150
200
250
300
350
temperature

Label Data Points

Use the text() function to add text labels at any position on the plot.

The position of the text is specified by the pos argument. Values of 1, 2, 3 and 4, respectively places the text below, to the left of, above and to the right of the specified coordinates.

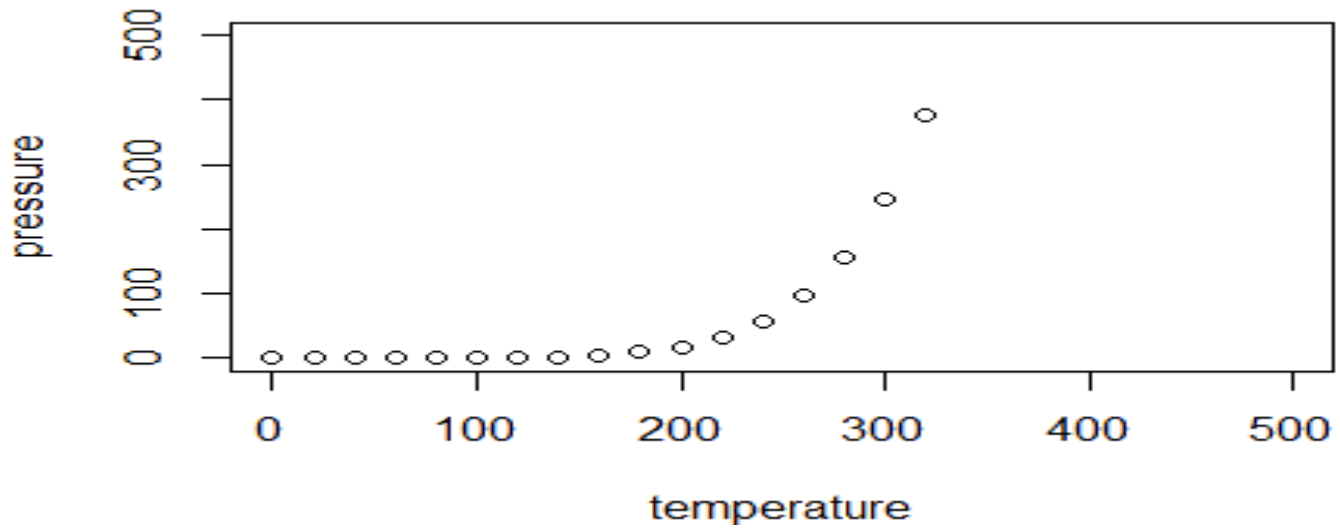
```
# Add text labels above the coordinates
plot(pressure, pch=19, col="red")
text(pressure, labels=pressure$pressure, cex=0.7, pos=3, col="blue")
```



Set Axis Limits

By default, the plot() function works out the best size and scale of each axis to fit the plotting area. However, you can set the limits of each axis quite easily using xlim and ylim arguments.

```
# Change the axis limits so that the x-axis and y-axis ranges from 0 to 500
plot(pressure, ylim=c(0,500), xlim=c(0,500))
```



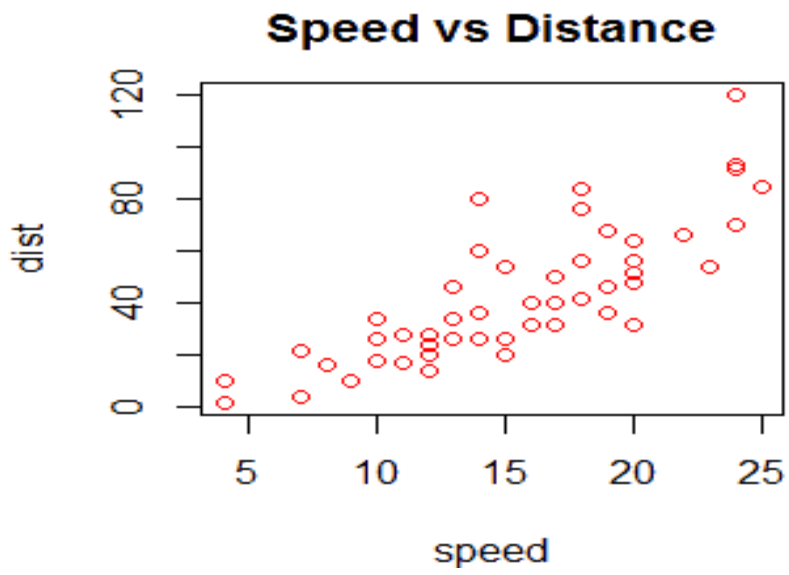
Display Multiple Plots on a Single Page

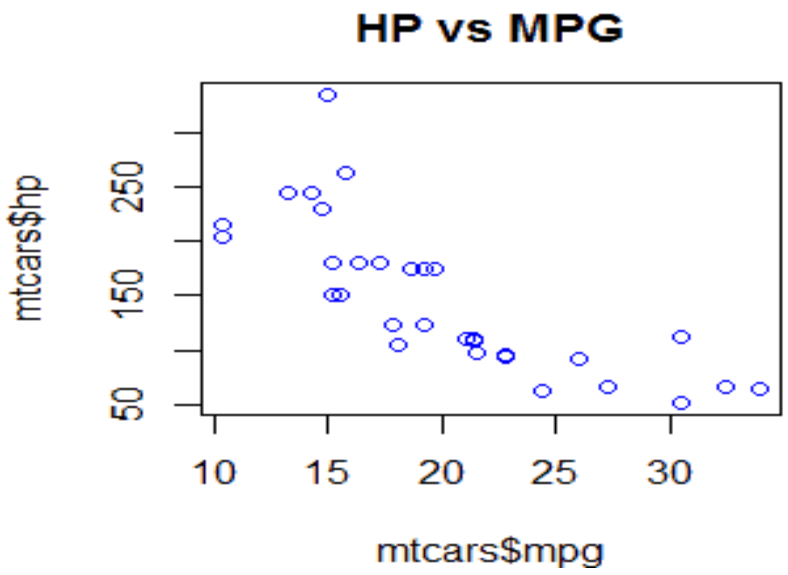
By using the mfrow graphics parameter, you can display multiple plots on the same graphics page.

To use this parameter, you need to pass a two-element vector, specifying the number of rows and columns. Then fill each cell in the matrix by repeatedly calling plot.

For example, mfrow=c(1, 2) creates two side by side plots.

```
par(mfrow = c(1, 2))
plot(cars, main="Speed vs Distance", col="red")
plot(mtcars$mpg, mtcars$hp, main="HP vs MPG", col="blue")
```


Speed vs Distance
dist
0
40
80
120
5
10
15
20
25
speed

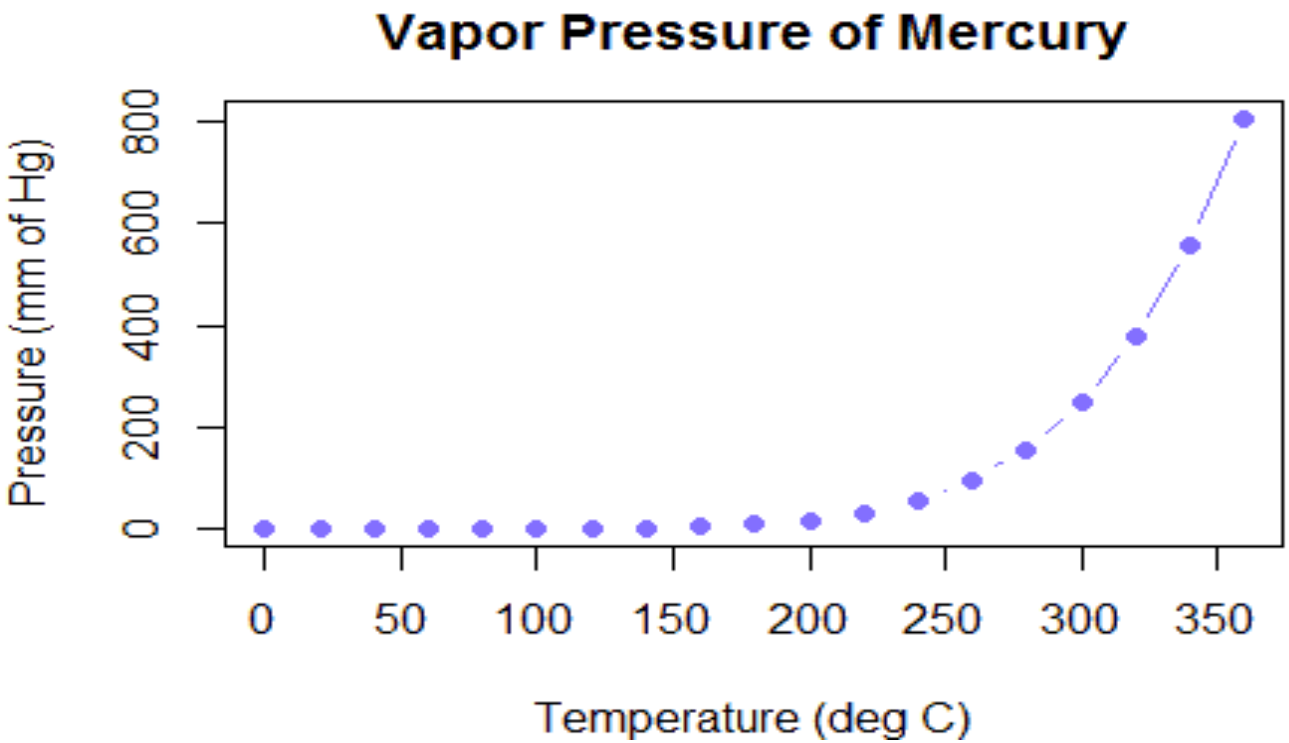

HP vs MPG
mtcars$hp
50
150
250
10
15
20
25
30
mtcars$mpg

Save a Plot to an Image File

To save a plot to an image file, you have to do three things in sequence:

Call a function to open a new graphics file, such as png(), jpg() or pdf().
Call plot() to generate the graphics image.
Call dev.off() to close the graphics file.

```
# Save a plot as a png file
png(filename="myPlot.png", width=648, height=432)
plot(pressure, col="slateblue1", pch=19, type="b",
    main = "Vapor Pressure of Mercury",
    xlab = "Temperature (deg C)",
    ylab = "Pressure (mm of Hg)")
dev.off()
```


Vapor Pressure of Mercury
Pressure (mm of Hg)
0
200
400
600
800
0
50
100
150
200
250
300
350
Temperature (deg C)

Summary Statistics

- Mean
  - `mean(D$wg)`
    - `[1] NA`
  - It doesn't know what to do with the missing values.
  - `mean(D$wg, na.rm=TRUE)`
    - `[1] 16.62016`

Summary Statistics

Median

```
median(D$wg,na.rm=TRUE)
  [1] 15
```

Quantiles

```
quantile(D$wg,c(.25,.5,.75),na.rm=TRUE)
    25% 50% 75%
    8  15  20
```

Summary

```
summary(D$wg)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  0.00    8.00   15.00   16.62   20.00   70.00  132.00
```

Summary Statistics

Range

```
range(D$wg,na.rm=TRUE)
```

IQR

```
IQR(D$wg,na.rm=TRUE)
```

Standard Deviation

```
sd(D$wg,na.rm=TRUE)
```

Thank You
Question ?