R - programming language

print (" ")
→ Comment

name = "He"
x ← 42 (assigning) - leftward operators.

Datatypes:

numeric
var1 ← 3.14

var2 ← 88L
integer

text
var3 ← "hello"

Text → string
Single / double quotes

Add → + − * /

^ (or) ** (Power)
%%. → remainder div
%./% → int division

Boolean
x > 20 → FALSE / TRUE

num ← 15
val ← num - 6

Math tasks
min( )
max( )

Relational
>, <, >=, <=, ==, !=

User Input
input = readLines ("stdin") → vector type! eg strings
print (input [2]) → 2nd line!

By default → String

Taking ints
x = as.integer(x);

---

x ← "hello"
print(x)
cat(x)

x ← "hello \n there"
print → didn't show line break
cat → shows line break.

if (x > 10)
{
}
else

else if(..)
{
}

Logical operators
&
|
!

Switch statement

* Test against values lpsk – make code much shorter – instead of else if

```
num ← 3 :
result ← switch (num, "one", "Two", "Three", "Four") → Three !
print (result) → print index
```

(or)

```
result ← switch (x, "a" = "one", "b" = "two", "c" = "Three", "d" = "four")
print (result)
```

Loops:

```
while (____) {
}
```

```
for (x in 1:10) {
    print(x)
}
```

→ iterate over lpsk & vectors.

break → breaks loop

next → skips loop

function

```
pow ← function (x,y) {
    result ← x ^y
    print (result)
}
```

```
pow(2,5)
pow(8,3)
```

default parameter values

```
pow ← function (x, y=2){
    result ← x ^y
    print (result)
}
```

Data Structures

→ paste → combine multiple strings into one

→ paste (k1, k2, k3, sep="_") → with seperator.

strings

```
tolower (kxk)
toupper (kxk)
```

```
kxk = "Sololearn"
```

Substr (txt, 2, 6); txt ← "some text"

↓ substring ↳ ome. t nchar (txt);

Vectors

* Sequence of elements of same data type — seperated by commas

'c function'

names ← c("James", "Amy", "John")

names [2] → Index starts from 1 not 0

ages ← c ("James" = 18, "Amy" = 14, "John" = 64)

ages ["James"]

Negative Indexing — skips that index

num ← c(1, 2, 3)

num [-3] → skips 3 prints 1 and 2

Vector functions

length()
sum()
sort()

x ← 3 : 9 (3, 4, 5, 6, 7, 8, 9)

seq (1, 10, by = 2) → sequence!

Condition

→ print (x [x < 50]) select elements less than 50

Vector arithmetic

v1 ← c(2, 6, 1, 5)
v2 ← c(5, 3, 4, 8)

$v_1 + v_2$
$v_1 - v_2$
$v_1 * v_2$
v_1 / v_2

mean (v)
median (v)

Lists

* List → hold different types of data

Keyword: list

x ← list ("James", c(2,4,8), 42, "Bob")

Alternative to

x ← list ("name" = "James", "age" = 42, "gendoor = 1")

x[1] ⟶ $name
[1] "James"

(or) – directly

x $name instead of x[name]

Add an element:

x[["country"]] ← "India" (or) x["country"] = "India"

x $ age same as x[["age"]]

List operations

merge

x ← list ("A", "B", "C")
y ← list ("D", "E")
z ← c(x, y)

Convert list to vector – to perform arithmetic operations

x ← list (4, 2, 11)
y ← unlist(x)
print (sort(y))
print (mean(y))

Matrix

2D – rows & columns – vector with addition dim

Keyword: matrix

nrow, ncol

matrix (c (1,2,3,4,5,6), nrow = 2, ncol = 3)

Default: columnwise

Print matrix

print (x[1,3])

1st row 3rd column

print (x[1,]) → First row full!

Operations

t() → transpose

Dataframes

vector : same type – 1d

list – vectors (simplest) – different types of elements

Matrices – like vectors – 2d

more than 2D – data frames

data frame – table

* each column has a name – has any data types

x ← data.frame ("id" = 1:2, "name" = c ("James", "Amy"), "age" = c (42, 18))

↓

3 columns

Access: double square brackets / $ operator

x [[2]] · (or) x [["name"]]

x $ name

double brackets: indicates – don't show column name.

x ["name"]

frame

1) James

2) Amy

x [[2, "name"]] → Select 2nd index from name

As columns of dataframes are vectors – use sum, mean, median

eg: age column

print (mean (x $ age))

factors

Dataframe of text column – R treats that column as Categorical data & creates factors on it.

Dataframe operation

Add new column – Assign to a vector

x $ country ← c ("USA", "Italy")

Filter rows based on condition

print (x [x $ age > 21,])

Using subset - Filtering

Subset (x, age > 21)

→ factors – variables in R – limited no. of different values

→ month names, weekdays → useful in statistical analysis

```
gender ← factor (c ("Male", "Female", "Male"))
print (gender)
```

each value of vector has its own number

```
table (gender)
x $ gender ← gender
print (x)
```

→ Female 1, Male 2

why: F < M (sort)

1 2

Import data

```
data ← read.csv ("demo.csv")
print (data [1, ])
```

11 columns - mtcars

mpg – miles per gallon (US)
cyl – Number of cylinders
disp – displacement (cu.in)
hp – Gross horsepower
drat – Rear axle ratio
wt – weight (1000 lbs)
qsec – 1/4 mile time
vs – Engine (0=V-Shaped, 1 = Straight
am – Transmission (0 – automatic, 1 – manual)
gear – No. of forward gears
carb – No. of carburetors

```
mt cars [1:5, ]
Summary (mt Cars)
```

↓

Min
1st qu
median
mean
3rd qu
max

Variance – measure of spread in a data set

Standard deviation – measure of dispersion of set of data from its mean – higher the dispersion – greater the S.D

```
Var ()
Sd ()
Var (mtcars $ mpg)
Sd (mt Cars $ mpg)
```

Sum (team $ height)

mtcars [mtcars $ hp == max (mtcars $ hp)] → which can corresponds to max hp

mtcars [mtcars $ mpg > 30,] → mpg value more than 30 rows

filter data – multiple condition

fastest car - automatic gear box

x ← mtcars [mtcars $ am == 0,]

x [x $ qsec == min (x $ qsec),]

Average qsec - that have hp > 100, gear < 5.

↓

get that column alone

x ← mtcars [mtcars $ hp > 100 &
mtcars $ gear < 5, 'qsec']

print (mean(x))

Correlation

* dependence btw multiple rows - used in data analytics.
* highly correlated - 0
* cor ()

res ← cor (mtcars)

res ← round (res, 2)

print (res)

Group data

Avg horse power - grouped by transmission type.

by (mtcars $ hp, mtcars $ am, mean)

- Apply the function
- Column to group the data
- function to apply

Sum of "wt" column grouped by "vs"

by (mtcars $ wt, mtcars $ vs, sum)

Group - Alternate way - Run aggregate functions

tapply (mtcars$hp, mtcars$am, mean)

Same pattern as "by()" - only diff - tapply - returns matrix
by - returns list object.

tapply (mtcars$hp, mtcars$cyl, max)

Cars with max hp, grouped by no. of cylinders

---

hp column - with > 4 gears

mtcars [mtcars$gear > 4, "hp"]

by (mtcars$mpg, mtcars$cyl, max)

---

data ← read.csv("people.csv")
data ← data [data$age > 18,] → all rows → age > 18
res ← mean (data$height) → take avg height of age > 18
print (res)

---

x ← mtcars [mtcars$mpg > 20,]
print (min (x$wt))

Titanic data

how Pclass - impacted on survival by taking mean of
age > = 18, grouped by Survived column.

x ← read.csv ("/usercode/files/titanic.csv")
y ← x [x$age > = 18]
y ← tapply (y$Pclass, y$Survived, mean)
print(y)

0	1
2.472	1.85

Plotting

png (file = "chart.png") → give own chart a name

plot () → create chart

main → chart title

xlab → x label

ylab → y label

plot (1:100, main = "my chart",
xlab = "x-axis",
ylab = "y-axis")

→ plot (x, y, xlab, ylab) → x & y coordinates seperately

x ← mtcars $wt

y ← mtcars $drat

line graph

plot (1:10, type = 'l')

plot (data, type = 'l')

Colour

plot (line1, type = 'l', col = 'blue')

multiple lines

for 1st line use plot

Remaining → lines() → Similar to plot

plot (line 1, type = 'l', col = 'blue')

lines (line 2, " , col = 'red')

Barcharts

barplot (mtcars$hp)

barplot (mtcars$hp, names.arg = rownames (mtcars))

horizontal barchat

barplot (mtcars$hp, horiz = TRUE)

barplot (data, names.arg = x, horiz = TRUE, col = "red")

piechart

x ← tapply (mtcars$hp, mtcars$gear, mean)

labels ← names (x)

png (file = "chart.png")

pie (x, label = labels, main = "Average HP by Gears");

x ← tapply (mtcars$wt, mtcars$vs, length)

pie(x)

"No. of cars for each 'vs' group"

Boxplot & histograms

boxplot → how distributed is the data.

boxplot (mtcars$mpg) → Show mean, min/max

avg value - black horizontal line

Histogram — frequency of each values (Continuous values)

hist (mtcars$hp)

similar to bars