An Introduction to R-Studio

With

Mr Fugu Data Science

(◕‿◕✿)

In [141]:
```
# Basic useful commands:

# get the current working directory:
getwd()

# set working directory:
setwd()
```

'/Users/zatoichi59'

In [1]:
```
# you can use R like a calculator:

factorial(5)
```

120

In [ ]:
```
# Assign Variables:
```

In [1]:
```
x<-4
y<- sqrt(7)

x*y

# Round to specified number of decimals
round(x*y,4)

# truncate: cutting of decimal
trunc(x*y)

# Floor:
floor(x*y)

# Ceiling:
ceiling(x*y)
```

10.5830052442584

10.583

10

10

11

In [2]:
```
# Replace variables:

old<-c(0,0,0,0,0)
old

old[3]<-'yay'
old
```

0 · 0 · 0 · 0 · 0

'0' · '0' · 'yay' · '0' · '0'

Vectors: basic syntax `c( , )`

- One of the palces R, shines; as well as matrices and data frames

In [18]:
```
my_firstVec <- c(-1,-3,0,9)
my_firstVec


# A vector will default to its weakest type:
new_vec<- c('Icecream',-1,0.5)

#Notice the strings around the digits
new_vec
```

-1 · -3 · 0 · 9

'Icecream' · '-1' · '0.5'

5 · 6 · 7

In [20]:
```
# Sequence
seq_vec<- 5:7
seq_vec

# sequence with range
sq_vec<-seq(5,9,by=.6)
sq_vec

# vector of specified length within your input range
sq_vec_len<-seq(2,8,length=9)
sq_vec_len
```

5 · 6 · 7

5 · 5.6 · 6.2 · 6.8 · 7.4 · 8 · 8.6

2 · 2.75 · 3.5 · 4.25 · 5 · 5.75 · 6.5 · 7.25 · 8

In [32]:
```
# Repetitions:

# think concatinating to end:
rep(c('I',"love",'data',"science"),2)

# repeats sequence in place
rep(c('I',"love",'data',"science"),c(1,2,3,3))

# -----------
```

'I' · 'love' · 'data' · 'science' · 'I' · 'love' · 'data' · 'science'

'I' · 'love' · 'love' · 'data' · 'data' · 'data' · 'science' · 'science' · 'science'

In [35]:
```
# Vector operations such as multiply and addition (term by term) can be
 done:

seq_vec*3

seq_vec+2*sq_vec_len

# you can access values by index: one at a time or many
seq_vec[1]

seq_vec[2:3]

sq_vec_len[c(1,3,6)]
```

15 · 18 · 21

9 · 11.5 · 14 · 13.5 · 16 · 18.5 · 18 · 20.5 · 23

5

6 · 7

2 · 3.5 · 5.75

In [42]:
```
# can do a Boolean comparison:
comp_vec<-seq_vec+2*sq_vec_len
comp_vec>16

#find values meeting boolean operation
comparison_vec<-comp_vec>16
comp_vec[comparison_vec]
```

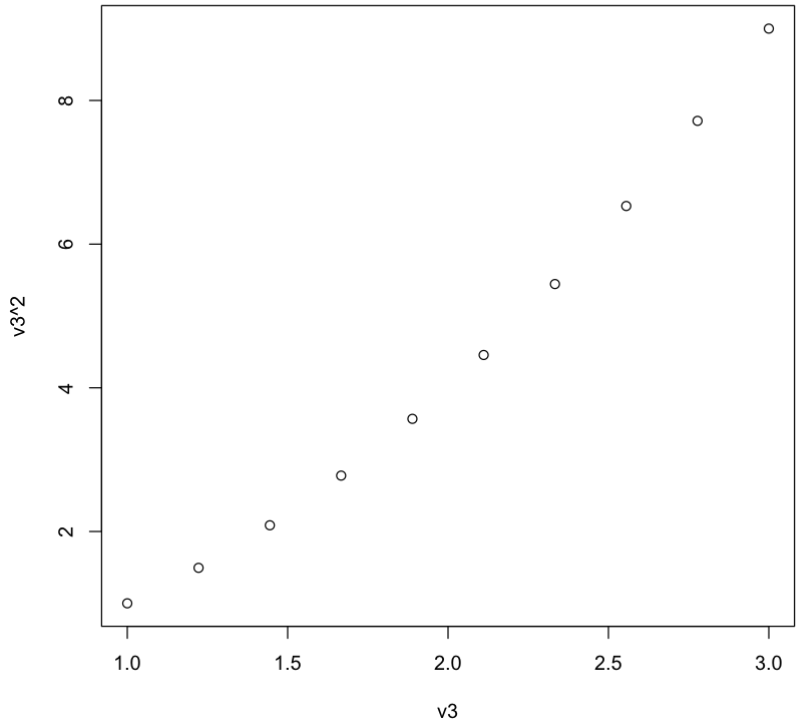
FALSE · FALSE · FALSE · FALSE · FALSE · TRUE · TRUE · TRUE · TRUE

18.5 · 18 · 20.5 · 23

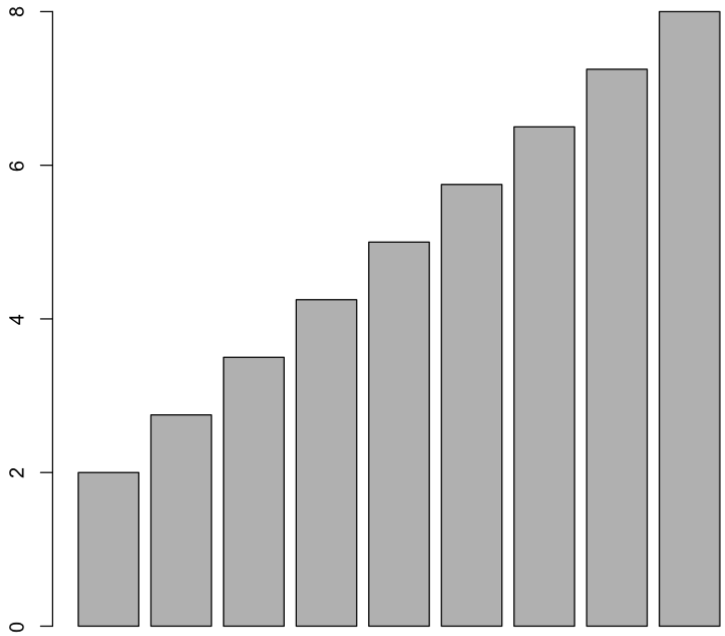
Plotting:

- Base R, has plotting functionality
  - Libraries such as `ggplot2` are often used for advanced features

In [138]:
```
v3<-seq(1,3,length=10)
plot(v3,v3^2)
```



In [140]:
```
# Simple Barplot:
barplot(sq_vec_len )
```



In [ ]:
```
# can do various other plots such as histograms etc..
```

We can write functions or equations using Latex:

this can be used to make literature or presentations look professional

encapsulate what you want to write within dollar signs: `$$something_you want $$`

$$\sum_{n=1}^{10} n^2$$

Probability & Statistics (*Built in functions*) Base R

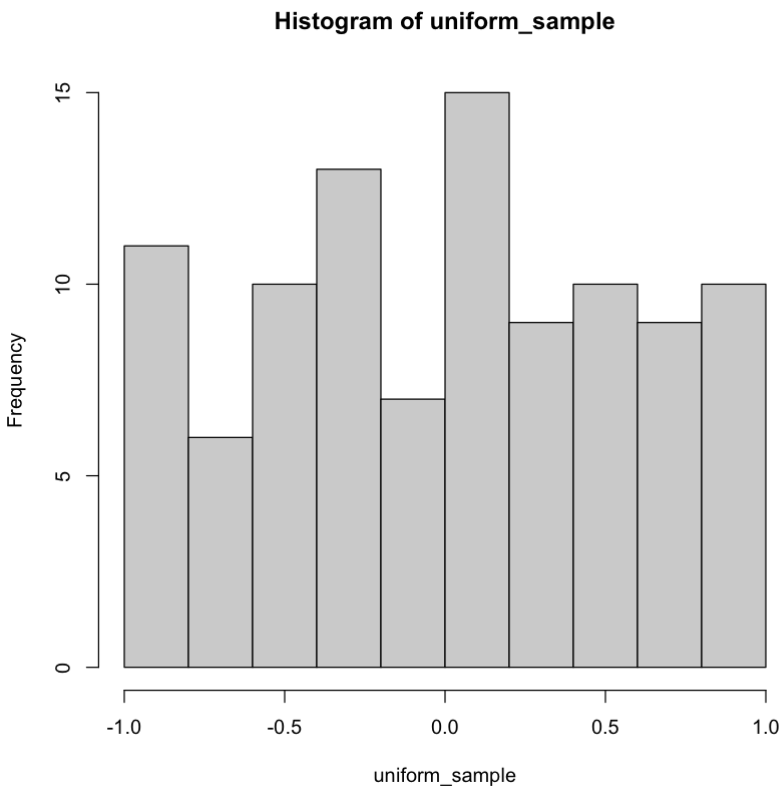
- `what if you know a function or have an idea but not what it does?`
  - for example we have `runif()` , but what does it do or how to use it?

well put a question mark in front and you will get the documentation: `?runif`

In [4]:
```
# The runif() function is part of a family of functions for (Uniform Dis
tribution)

uniform_sample <- runif(min = -1 ,max = 1 ,n = 100)
uniform_sample[1:10]
```

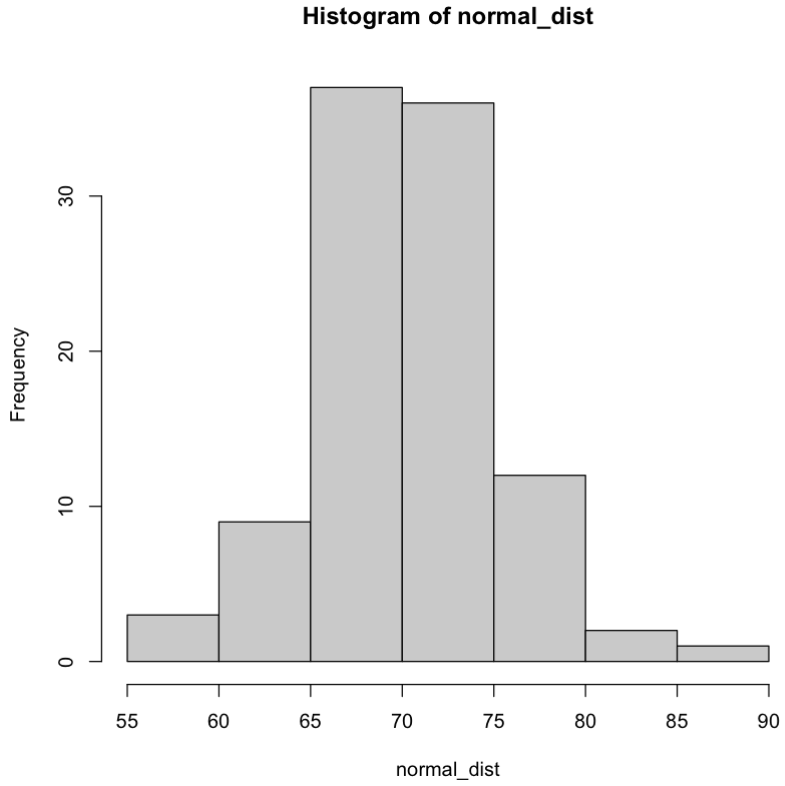
0.245535854250193 · 0.367100795265287 · -0.870561884250492 · 0.187113768886775 · -0.844769279938191 · 0.813514930661768 · 0.847346059512347 · 0.225478164386004 · 0.321629679761827 · 0.55576233798638

In [5]:
```
hist(x = uniform_sample)
```

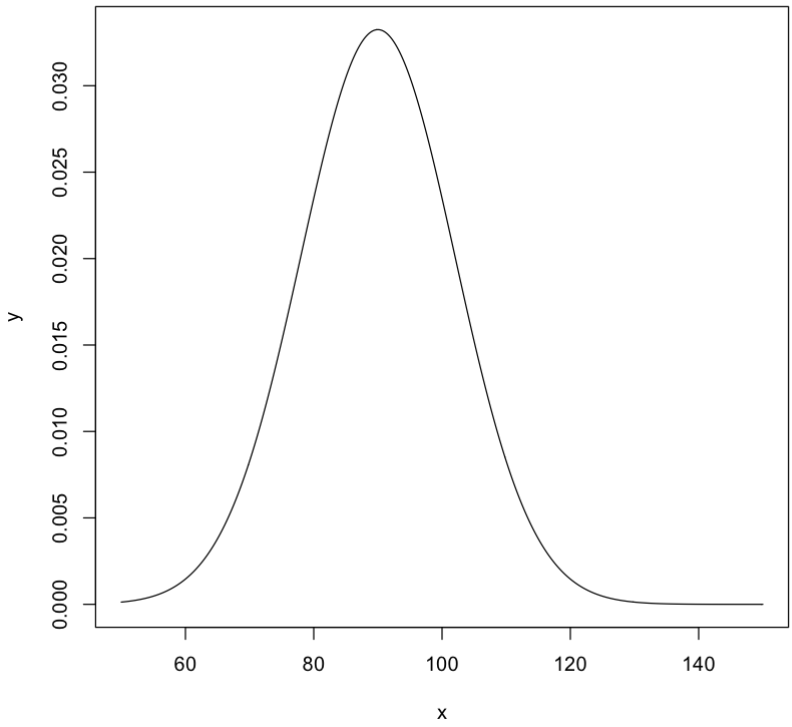


In [89]:
```
# Normal distribution: we have options if you do ?rnorm, and notice [q,
p,d]norm

# if you don't set a (seed), then your results will change each time cod
e is run
set.seed(1234)
rnorm(100,60,5)[3:10]


normal_dist<-rnorm(100,70,5)
hist(normal_dist)
```

65.4222058834153 · 48.2715114868533 · 62.1456234440552 · 62.5302794607879 · 57.1263001993268 · 57.2668407210791 · 57.1777400045336 · 55.5498108547795



In [108]:
```
# Let's look at the actual distribution:
x<-seq(50,150,by=.2)
y<-dnorm(x,mean=90,sd=12)
plot(x,y,type = "l")
# ?plot
```



matrices:

In [113]:
```
# Pay Attention here:

dat<-c(2,4,6,8,11)
first_mat <- matrix(data =dat ,byrow =TRUE,nrow = 3,ncol = 2 )
first_mat
```

```
Warning message in matrix(data = dat, byrow = TRUE, nrow = 3, ncol =
2):
“data length [5] is not a sub-multiple or multiple of the number of row
s [3]”
```

A
matrix:
3 × 2 of
type dbl

2	4
6	8
11	2

In [137]:
```
# if you know the number of rows you want; R can figure columns
dta<-c(2,3,4,5,6,7,8,9,2,3,4,5,6,7,8,9)
second_mat<-matrix(dta,nrow = 4)
second_mat

# access elements within the matrix:

# row 1, column 2
second_mat[1,2]

# column 1:
second_mat[,1]

# range values for rows, and all columns
second_mat[1:3,]

# specific rows or columns:
second_mat[c(1,3),c(1,4)]

# find the dimensions of matrix
dim(second_mat)
```

A matrix: 4 ×
4 of type dbl

2	6	2	6
3	7	3	7
4	8	4	8
5	9	5	9

6

2 · 3 · 4 · 5

A matrix: 3 ×
4 of type dbl

2	6	2	6
3	7	3	7
4	8	4	8

A
matrix:
2 × 2
of type
dbl

2	6
4	8

4 · 4

In [136]:
```
# descriptive stats:

mean(x = dat)

median(dat)

sd(dat)
```

6.2

6

3.4928498393146

LIKE Share &

SUBscribe

Further Useful Help:

https://dss.princeton.edu/training/RStudio101.pdf (https://dss.princeton.edu/training/RStudio101.pdf)

https://sites.tufts.edu/datalab/files/2018/04/R_RStudio_Basics.pdf
(https://sites.tufts.edu/datalab/files/2018/04/R_RStudio_Basics.pdf)

https://support.rstudio.com/hc/en-us/articles/115010737148-Using-the-RStudio-Terminal
(https://support.rstudio.com/hc/en-us/articles/115010737148-Using-the-RStudio-Terminal)

https://sites.calvin.edu/scofield/courses/m143/materials/RcmdsFromClass.pdf
(https://sites.calvin.edu/scofield/courses/m143/materials/RcmdsFromClass.pdf)