Julia Programming Intermediate

[Julia Documentation (https://docs.julialang.org/en/v1/)](https://docs.julialang.org/en/v1/)

Version : 1.6.1

Created By Manthan Bhikadiya

Reference Video Tutorial by Abhishek Agrawal :

- [https://www.youtube.com/watch?v=lwj-1mclq0U (https://www.youtube.com/watch?v=lwj-1mclq0U)](https://www.youtube.com/watch?v=lwj-1mclq0U)

Topics :

- Functions
- Formatting Number and Strings in Julia
- Working With CSV Files
- Data Visualization
- Working with Database
- Calling Python Packages
- Machine learning Project In Julia

Functions

In [1]:

```
f() = println("Running the function") # single liner function , without argument
```

Out[1]:

```
f (generic function with 1 method)
```

In [2]:

```
f()
```

```
Running the function
```

In [3]:

```
f(x) = x + x  # single argument function
```

Out[3]:

```
f (generic function with 2 methods)
```

In [4]:

```
println( f(2) )
println( f(2.4) )
```

```
4
4.8
```

In [5]:

```
f(x,y) = x*3 - y*2
```

Out[5]:

```
f (generic function with 3 methods)
```

In [6]:

```
f(3,2) # x=3,y=2 => 3*3 - 2*2 => 9-4 => 5
```

Out[6]:

```
5
```

In [7]:

```
# standard Function

function multiply(x,y)
    return (x*y) + (x+y)
end
```

Out[7]:

```
multiply (generic function with 1 method)
```

In [8]:

```
multiply(2,3) # x=2,y=3 => (2*3) + (2+3) => 6 + 5 => 11
```

Out[8]:

```
11
```

In [9]:

```
# simple function to check wheather a number is prime or not
# return true if number is prime otherwise false.

function is_prime(number)
    if number == 1
        return "Non-Prime (special case)"
    else
        for i in 2:number-1
            if number%i == 0
                return "Non-Prime"
            end
        end
    end
    return "Prime"
end
```

Out[9]:

```
is_prime (generic function with 1 method)
```

In [10]:

```
is_prime(7)
```

Out[10]:

```
"Prime"
```

FOR MORE CHECKOUT OFFICIAL DOCUMENTATION :
Function (https://docs.julialang.org/en/v1/base/base/#function)

Formatting Numbers and String

In [11]:

```
using Printf
```

In [12]:

```
name = "Manthan"
```

Out[12]:

```
"Manthan"
```

In [13]:

```
# if you don't want to print the name in the console and just want to store it in variable
# ; semicolon at the end of statement.

name = "Manthan" ;
```

In [14]:

```
# whatever write with @ is called macro in julia

# printf macro

@printf("Hello %s",name)
```

```
Hello Manthan
```

In [15]:

```
# string based output
# basically it output the string within quotes ("")

# sprintf macro
@sprintf("Hello %s",name)
```

Out[15]:

```
"Hello Manthan"
```

In [16]:

```
# for character

ch = "i"

@printf("character : %c",ch)
@sprintf("character : %c",ch)
```

```
character : i
```

Out[16]:

```
"character : i"
```

In [17]:

```
# more example

x = 100

@printf("Value of x is %d",x)
```

```
Value of x is 100
```

In [18]:

```
y = 100.5
@printf("Value of y is %f",y)
```

```
Value of y is 100.500000
```

In [19]:

```
# float value use %d convert float to int.
@printf("Value of y is %d", y)

println() # just for new line

# reverse
@printf("Value of x is %f",x)
```

```
Value of y is 100
Value of x is 100.000000
```

In [20]:

```
# if you used %c with string then it will return first character of the string
@printf("Hello %c",name)

println()

# but for char we can use %s. It is totally valid.
@printf("character : %s",ch)
```

```
Hello M
character : i
```

In [21]:

```
z = 134.76366783882773

@printf("Number is %.2f \n",z) # showing only two points after decimal
@printf("Number is %e \n",z) # showing large number with %e
@printf("Number is %.3e",z) # 3 points after decimal
```

```
Number is 134.76
Number is 1.347637e+02
Number is 1.348e+02
```

In [22]:

```
z = 1234378718325736782368712

@printf("Number = %d \n",z) # very large number represent it wil %e
@printf("Number = %e \n",z) # by default show 6 points after decimal
@printf("number = %.2e \n",z) # 2 points after decimal
```

```
Number = 1234378718325736782368712
Number = 1.234379e+24
number = 1.23e+24
```

- %s - **String**
- %f - **Floating values**
- %d - **Integer Number**
- %c - **Character**
- %e - **Show large number in Exponent Form**

FOR MORE CHECKOUT GEEKS FOR GEEKS BLOG :
String & Number Formatting (https://www.geeksforgeeks.org/format-specifiers-in-julia/#:~:text=Julia%20contains%20a%20package%20Printf,%22%2C%20args)

Working With CSV Files

In [23]:

```
# if you don't have CSV Installed then run below script first
# using Pkg
# Pkg.add("CSV")
```

In [24]:

```
# another package is neccessary for read csv.
# it convert csv file to DataFrames

# using Pkg
# Pkg.add("DataFrames")
```

In [25]:

```
using CSV # import csv package
```

In [26]:

```
using DataFrames
```

```
┌ Info: Precompiling DataFrames [a93c6f00-e57d-5684-b7b6-d8193f3e46c0]
└ @ Base loading.jl:1317
```

In [27]:

```
iris = CSV.read("Iris.csv",DataFrame)
```

Out[27]:

150 rows × 6 columns

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
	Int64	Float64	Float64	Float64	Float64	String
1	1	5.1	3.5	1.4	0.2	Iris-setosa
2	2	4.9	3.0	1.4	0.2	Iris-setosa
3	3	4.7	3.2	1.3	0.2	Iris-setosa
4	4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	5.0	3.6	1.4	0.2	Iris-setosa
6	6	5.4	3.9	1.7	0.4	Iris-setosa
7	7	4.6	3.4	1.4	0.3	Iris-setosa
8	8	5.0	3.4	1.5	0.2	Iris-setosa
9	9	4.4	2.9	1.4	0.2	Iris-setosa
10	10	4.9	3.1	1.5	0.1	Iris-setosa
11	11	5.4	3.7	1.5	0.2	Iris-setosa
12	12	4.8	3.4	1.6	0.2	Iris-setosa
13	13	4.8	3.0	1.4	0.1	Iris-setosa
14	14	4.3	3.0	1.1	0.1	Iris-setosa
15	15	5.8	4.0	1.2	0.2	Iris-setosa
16	16	5.7	4.4	1.5	0.4	Iris-setosa
17	17	5.4	3.9	1.3	0.4	Iris-setosa
18	18	5.1	3.5	1.4	0.3	Iris-setosa
19	19	5.7	3.8	1.7	0.3	Iris-setosa
20	20	5.1	3.8	1.5	0.3	Iris-setosa
21	21	5.4	3.4	1.7	0.2	Iris-setosa
22	22	5.1	3.7	1.5	0.4	Iris-setosa
23	23	4.6	3.6	1.0	0.2	Iris-setosa
24	24	5.1	3.3	1.7	0.5	Iris-setosa
25	25	4.8	3.4	1.9	0.2	Iris-setosa
26	26	5.0	3.0	1.6	0.2	Iris-setosa
27	27	5.0	3.4	1.6	0.4	Iris-setosa
28	28	5.2	3.5	1.5	0.2	Iris-setosa
29	29	5.2	3.4	1.4	0.2	Iris-setosa
30	30	4.7	3.2	1.6	0.2	Iris-setosa
⋮	⋮	⋮	⋮	⋮	⋮	⋮

In [28]:

```
typeof(iris)
```

Out[28]:

```
DataFrame
```

In [29]:

```
names(iris) # names of columns
```

Out[29]:

```
6-element Vector{String}:
 "Id"
 "SepalLengthCm"
 "SepalWidthCm"
 "PetalLengthCm"
 "PetalWidthCm"
 "Species"
```

In [30]:

```
size(iris) # shape of data
```

Out[30]:

```
(150, 6)
```

In [31]:

```
first(iris, 5) # first five rows of dataset
```

Out[31]:

5 rows × 6 columns

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
	Int64	**Float64**	**Float64**	**Float64**	**Float64**	**String**
1	1	5.1	3.5	1.4	0.2	Iris-setosa
2	2	4.9	3.0	1.4	0.2	Iris-setosa
3	3	4.7	3.2	1.3	0.2	Iris-setosa
4	4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	5.0	3.6	1.4	0.2	Iris-setosa

In [32]:

```
last(iris, 5) # last fivee rows of dataset
```

Out[32]:

5 rows × 6 columns

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
	Int64	Float64	Float64	Float64	Float64	String
1	146	6.7	3.0	5.2	2.3	Iris-virginica
2	147	6.3	2.5	5.0	1.9	Iris-virginica
3	148	6.5	3.0	5.2	2.0	Iris-virginica
4	149	6.2	3.4	5.4	2.3	Iris-virginica
5	150	5.9	3.0	5.1	1.8	Iris-virginica

In [33]:

```
# some stats about data

describe(iris)
```

Out[33]:

6 rows × 7 columns

	variable	mean	min	median	max	nmissing	eltype
	Symbol	Union…	Any	Union…	Any	Int64	DataType
1	Id	75.5	1	75.5	150	0	Int64
2	SepalLengthCm	5.84333	4.3	5.8	7.9	0	Float64
3	SepalWidthCm	3.054	2.0	3.0	4.4	0	Float64
4	PetalLengthCm	3.75867	1.0	4.35	6.9	0	Float64
5	PetalWidthCm	1.19867	0.1	1.3	2.5	0	Float64
6	Species		Iris-setosa		Iris-virginica	0	String

In [34]:

```
# accessing specific columns
iris.Species
```

Out[34]:

```
150-element PooledArrays.PooledVector{String, UInt32, Vector{UInt32}}:
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 ⋮
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
```

In [35]:

```
# accessing particular rows and column (Indexing)
iris[:,2] # SepalLength
```

Out[35]:

```
150-element Vector{Float64}:
 5.1
 4.9
 4.7
 4.6
 5.0
 5.4
 4.6
 5.0
 4.4
 4.9
 5.4
 4.8
 4.8
 ⋮
 6.0
 6.9
 6.7
 6.9
 5.8
 6.8
 6.7
 6.7
 6.3
 6.5
 6.2
 5.9
```

In [36]:

```
iris[:,[2,3,4,5]] # all columns except Id (first index)
```

Out[36]:

150 rows × 4 columns

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
	Float64	Float64	Float64	Float64
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2
6	5.4	3.9	1.7	0.4
7	4.6	3.4	1.4	0.3
8	5.0	3.4	1.5	0.2
9	4.4	2.9	1.4	0.2
10	4.9	3.1	1.5	0.1
11	5.4	3.7	1.5	0.2
12	4.8	3.4	1.6	0.2
13	4.8	3.0	1.4	0.1
14	4.3	3.0	1.1	0.1
15	5.8	4.0	1.2	0.2
16	5.7	4.4	1.5	0.4
17	5.4	3.9	1.3	0.4
18	5.1	3.5	1.4	0.3
19	5.7	3.8	1.7	0.3
20	5.1	3.8	1.5	0.3
21	5.4	3.4	1.7	0.2
22	5.1	3.7	1.5	0.4
23	4.6	3.6	1.0	0.2
24	5.1	3.3	1.7	0.5
25	4.8	3.4	1.9	0.2
26	5.0	3.0	1.6	0.2
27	5.0	3.4	1.6	0.4
28	5.2	3.5	1.5	0.2
29	5.2	3.4	1.4	0.2
30	4.7	3.2	1.6	0.2
⋮	⋮	⋮	⋮	⋮

In [37]:

```
iris[1:5,:] # first five rows and all columns
```

Out[37]:

5 rows × 6 columns

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
	Int64	Float64	Float64	Float64	Float64	String
1	1	5.1	3.5	1.4	0.2	Iris-setosa
2	2	4.9	3.0	1.4	0.2	Iris-setosa
3	3	4.7	3.2	1.3	0.2	Iris-setosa
4	4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	5.0	3.6	1.4	0.2	Iris-setosa

In [38]:

```
iris[1:5,2:4] # first five rows and 3 columns
```

Out[38]:

5 rows × 3 columns

	SepalLengthCm	SepalWidthCm	PetalLengthCm
	Float64	Float64	Float64
1	5.1	3.5	1.4
2	4.9	3.0	1.4
3	4.7	3.2	1.3
4	4.6	3.1	1.5
5	5.0	3.6	1.4

FOR MORE CHECKOUT GEEKS FOR GEEKS BLOG :
CSV Package Documentation (https://csv.juliadata.org/stable/)

Data Visualization

In [39]:

```
# install plot package by using below commands
# using Pkg
# Pkd.add("Plots")
```

In [40]:

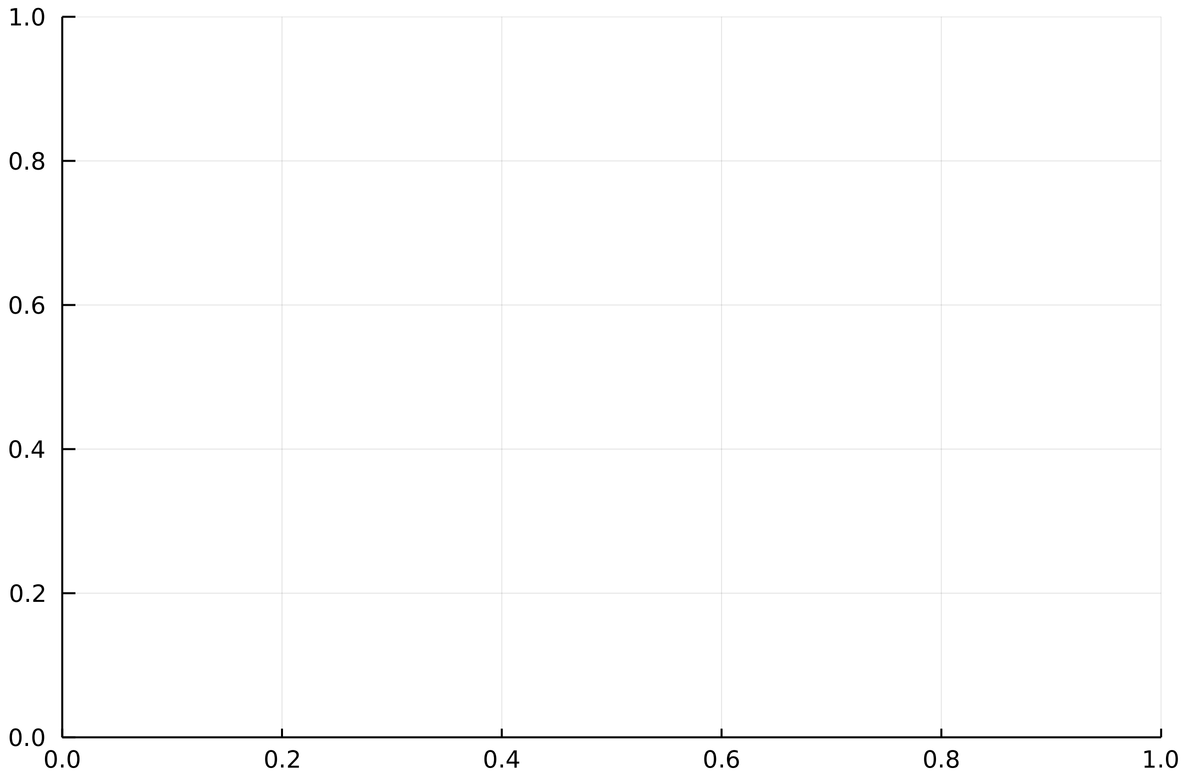
```
using Plots
```

```
┌ Info: Precompiling Plots [91a5bcdd-55d7-5caf-9e0b-520d859cae80]
└ @ Base loading.jl:1317
```

In [41]:

```
plot() # blank plot
```

Out[41]:



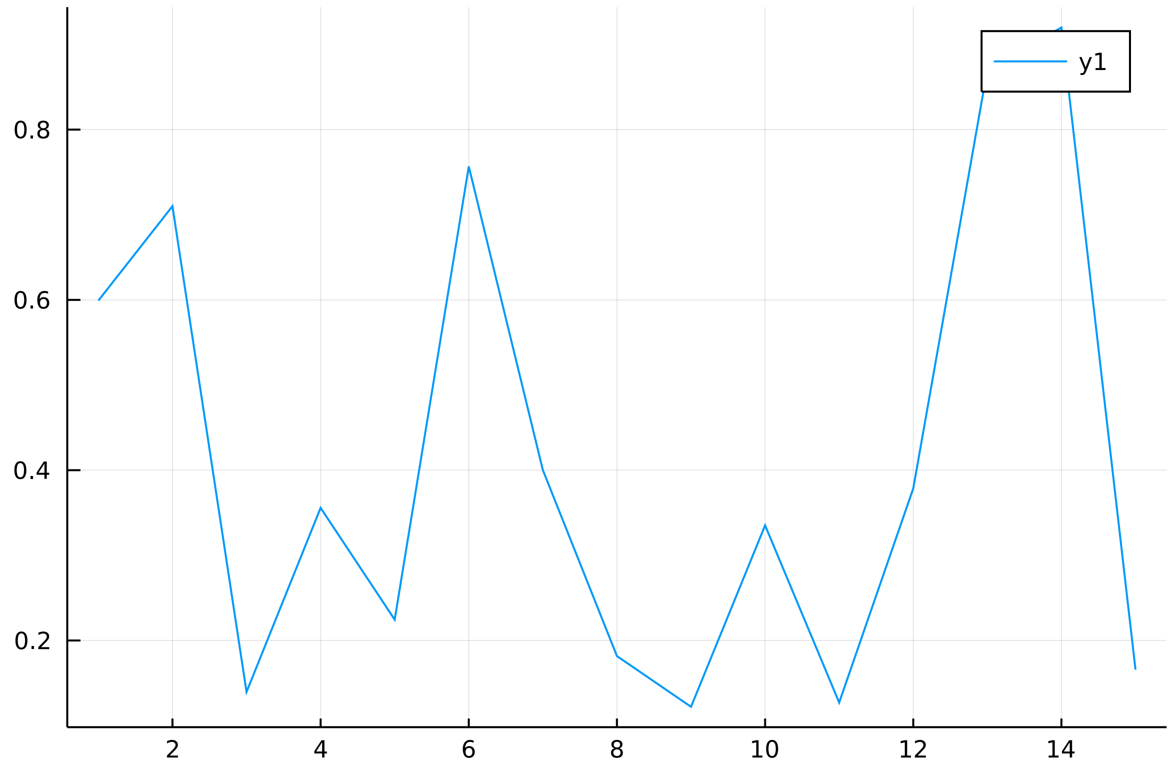
In [42]:

```
# make Data
# ignore print the data in notebook use ; at the end of command
x = 1:15;
y = rand(15);
```

In [43]:

```
plot(x,y)
```

Out[43]:



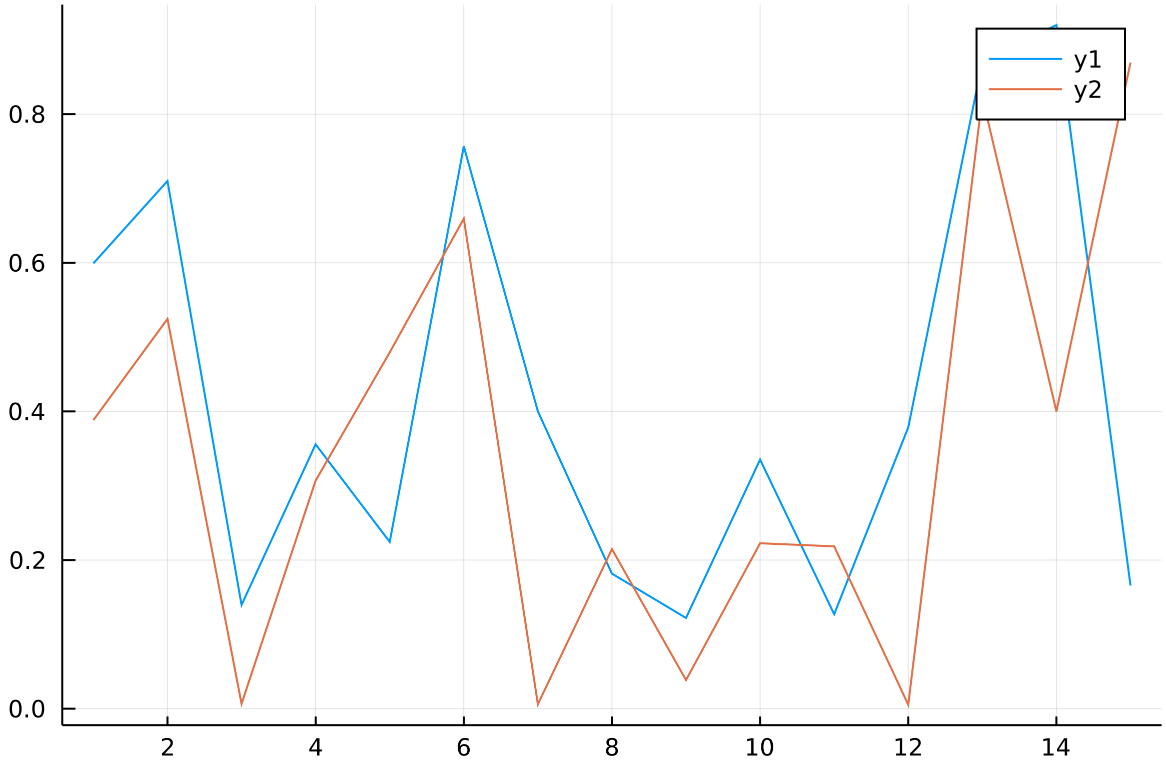
In [44]:

```
z = rand(15);
```

In [45]:

```
plot!(x,z) # add informaiton in past graph
```

Out[45]:

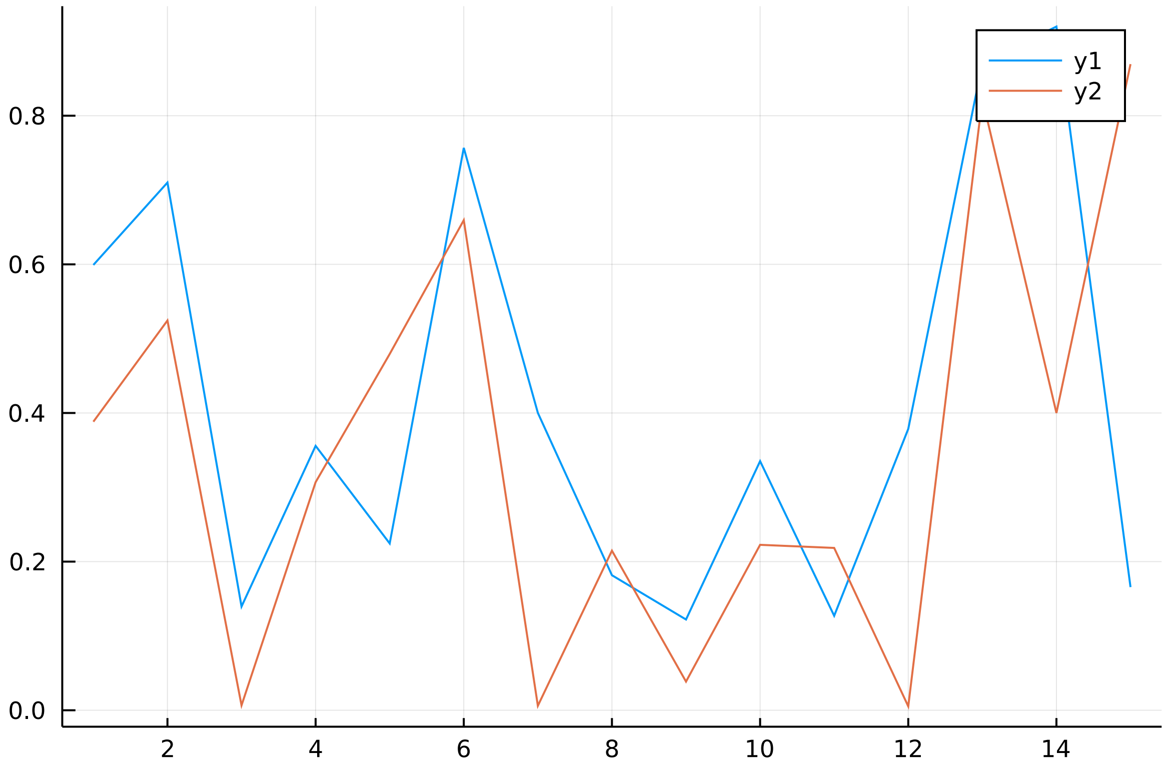


In [46]:

```
# alternative way to plot above graph

p = plot(x,y)
plot!(p,x,z)
```

Out[46]:

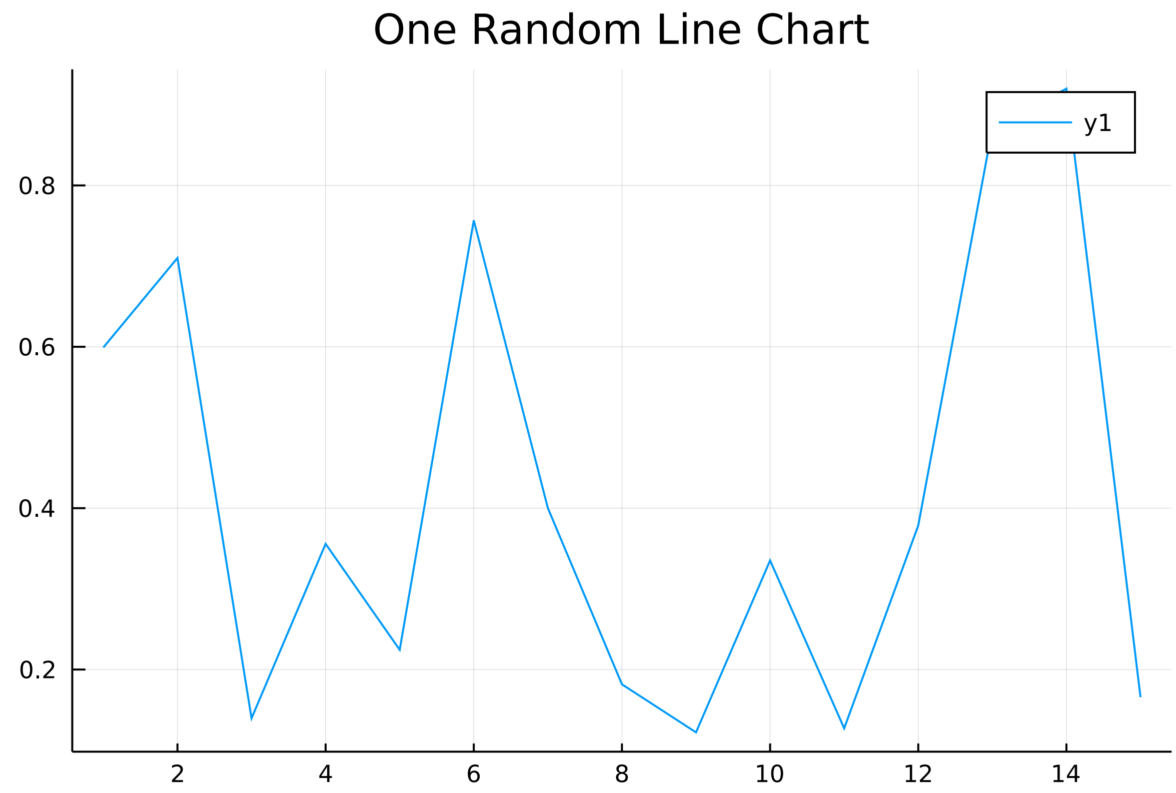


In [47]:

```
# add title in the graph

plot(x,y,title="One Random Line Chart")
```

Out[47]:



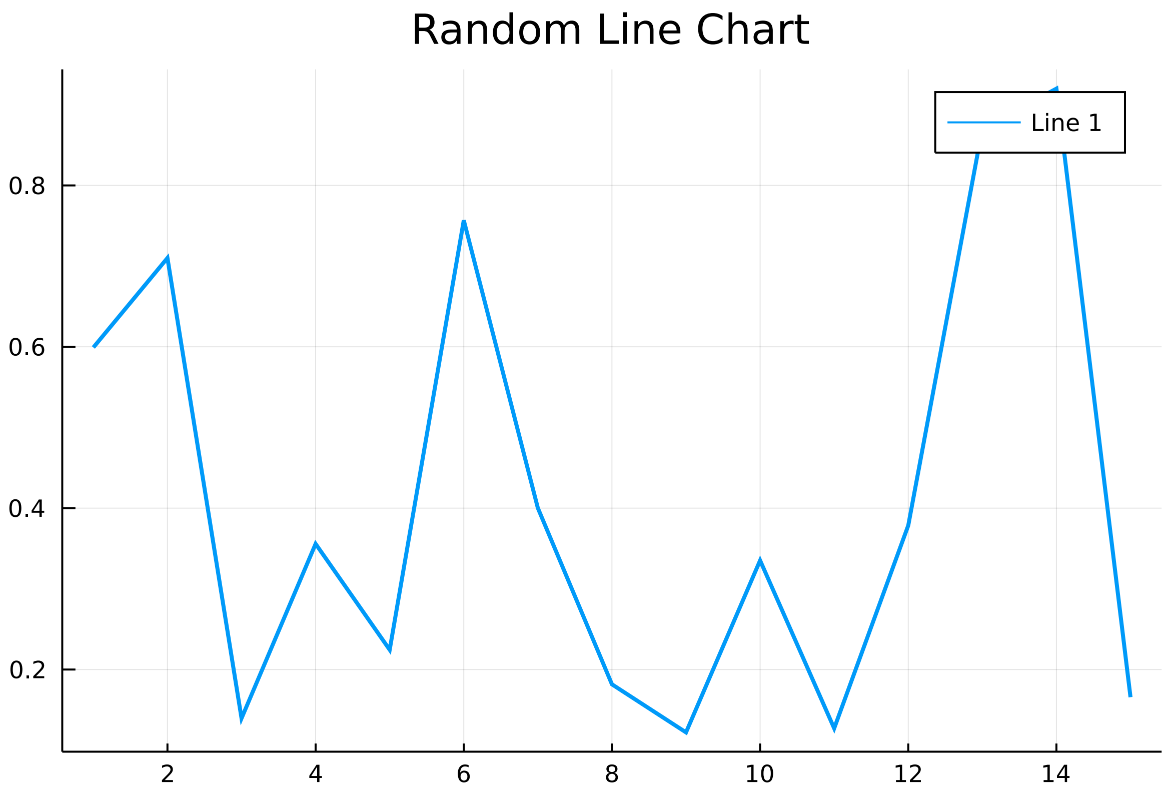
More Infromation About Plot() :- https://docs.juliaplots.org/latest/tutorial/ (https://docs.juliaplots.org/latest/tutorial/)

In [48]:

```
# adding label

plot(x, y , title="Random Line Chart" , label="Line 1" , lw = 2) # lw = linewidth
```

Out[48]:

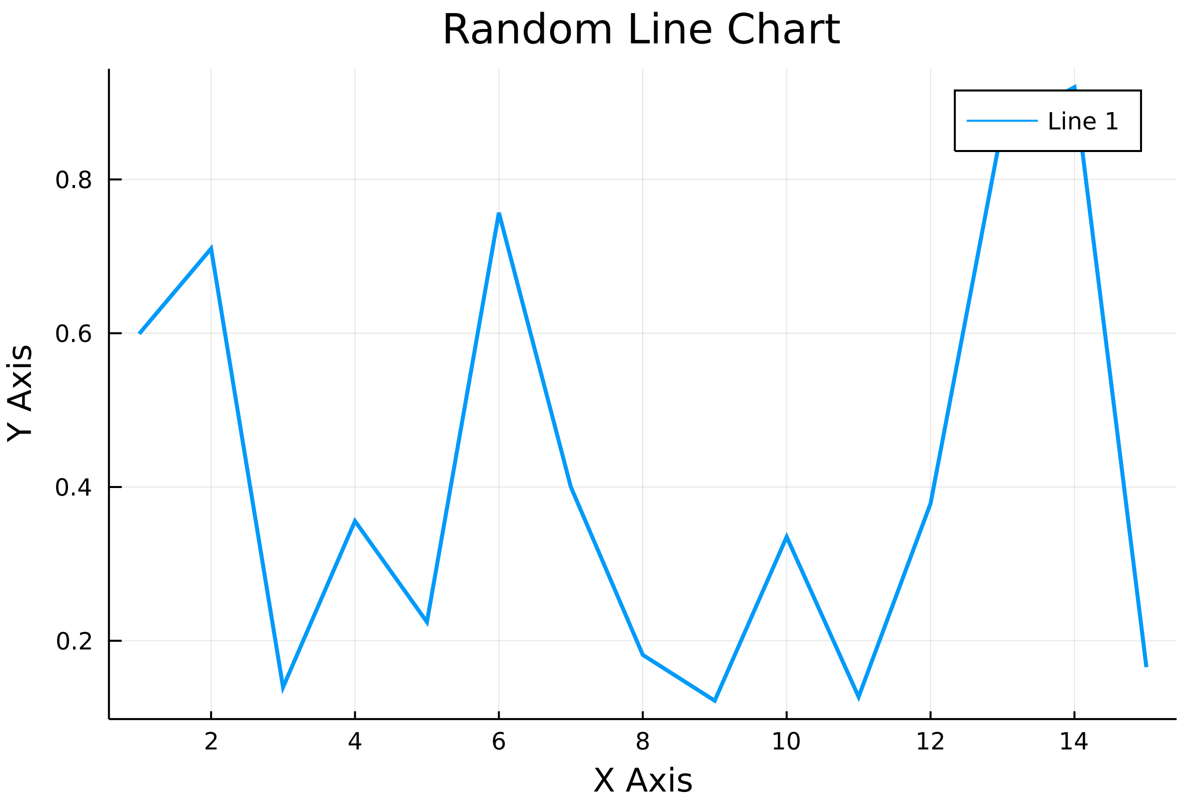


In [49]:

```
# x-axis label
# y-axis label

plot(x,y,title="Random Line Chart",label="Line 1",xlabel="X Axis",ylabel="Y Axis",lw = 2)
```

Out[49]:

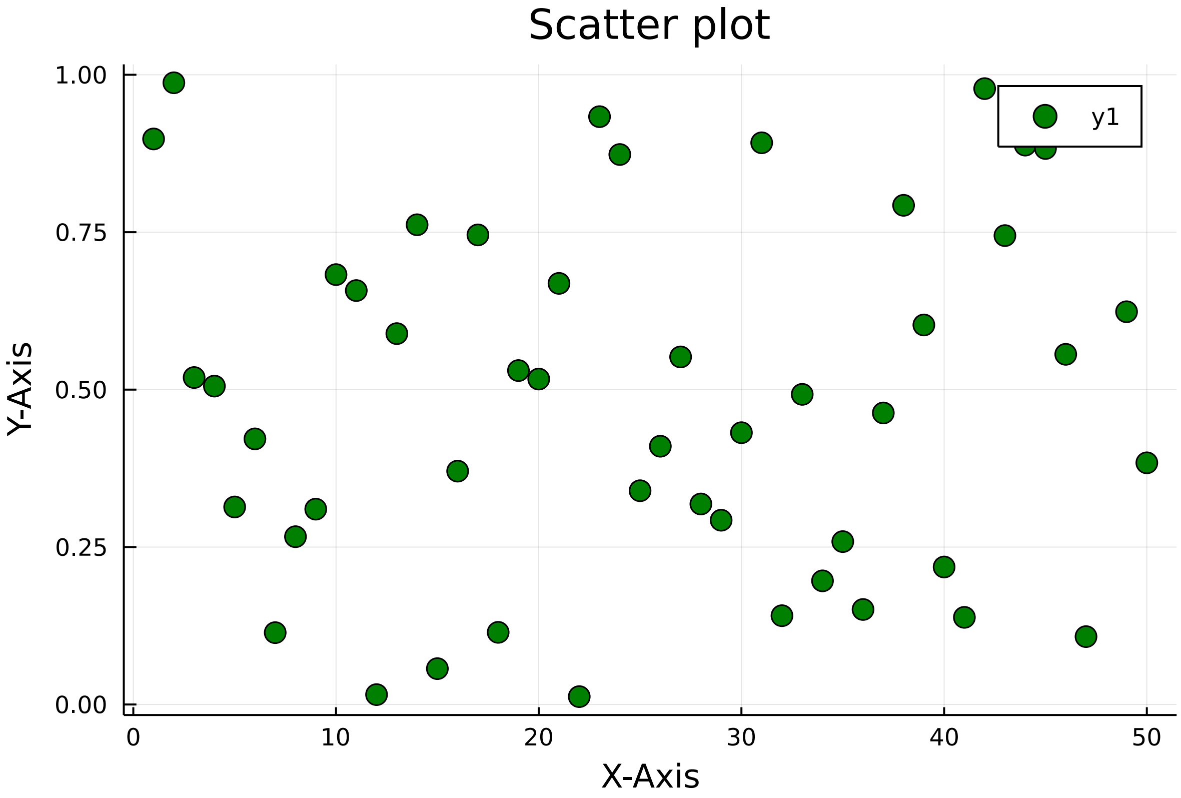


In [50]:

```
# scatter plot

scatter(1:50 , rand(50) ,markercolor="green",markersize = 6,title="Scatter plot",
    xlabel="X-Axis",ylabel="Y-Axis")
```

Out[50]:

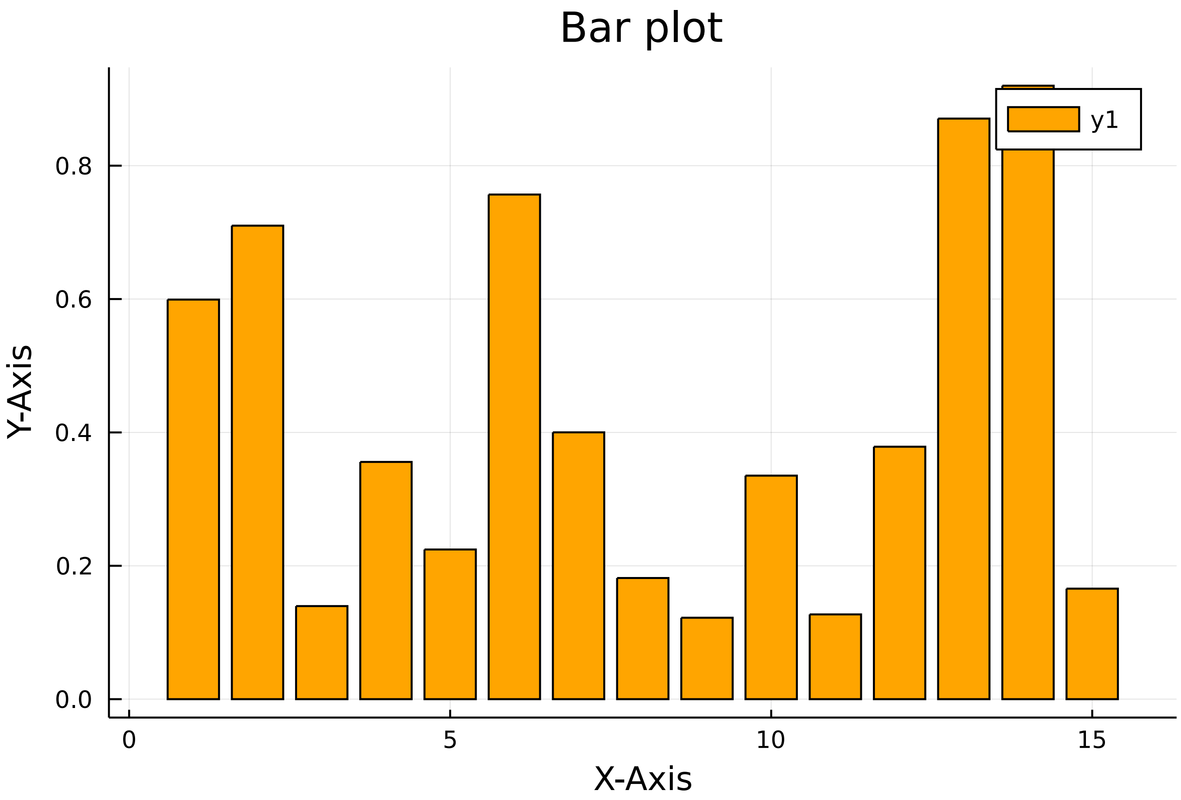


In [51]:

```
# bar plot

bar(x , y,title="Bar plot",xlabel="X-Axis",ylabel="Y-Axis",color="orange")
```

Out[51]:

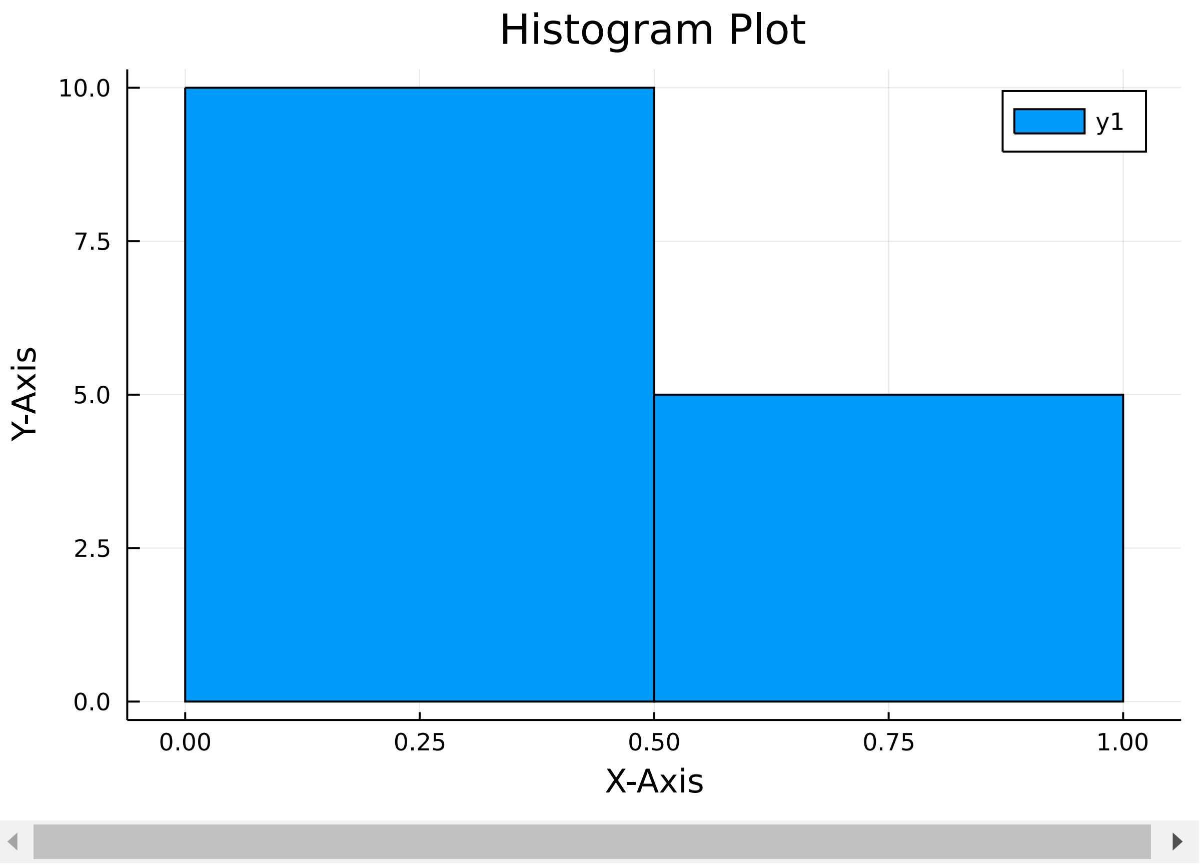


In [52]:

```
# histogram

histogram(y , title="Histogram Plot",xlabel="X-Axis",ylabel="Y-Axis")
```

Out[52]:



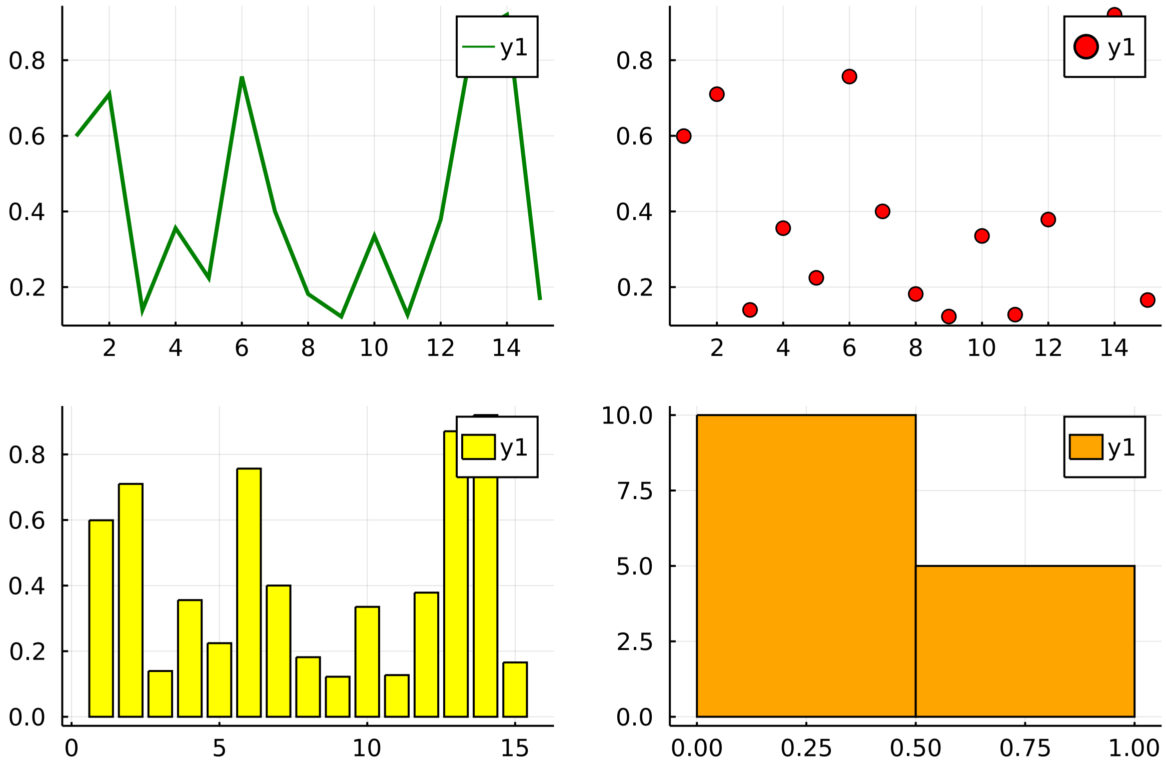
In [53]:

```
# combining all together

p1 = plot(x,y,color="green",lw = 2)
p2 = scatter(x,y,color="red")
p3 = bar(x,y , color = "yellow")
p4 = histogram(y , color="orange")

plot(p1,p2,p3,p4,layout = (2,2))
```

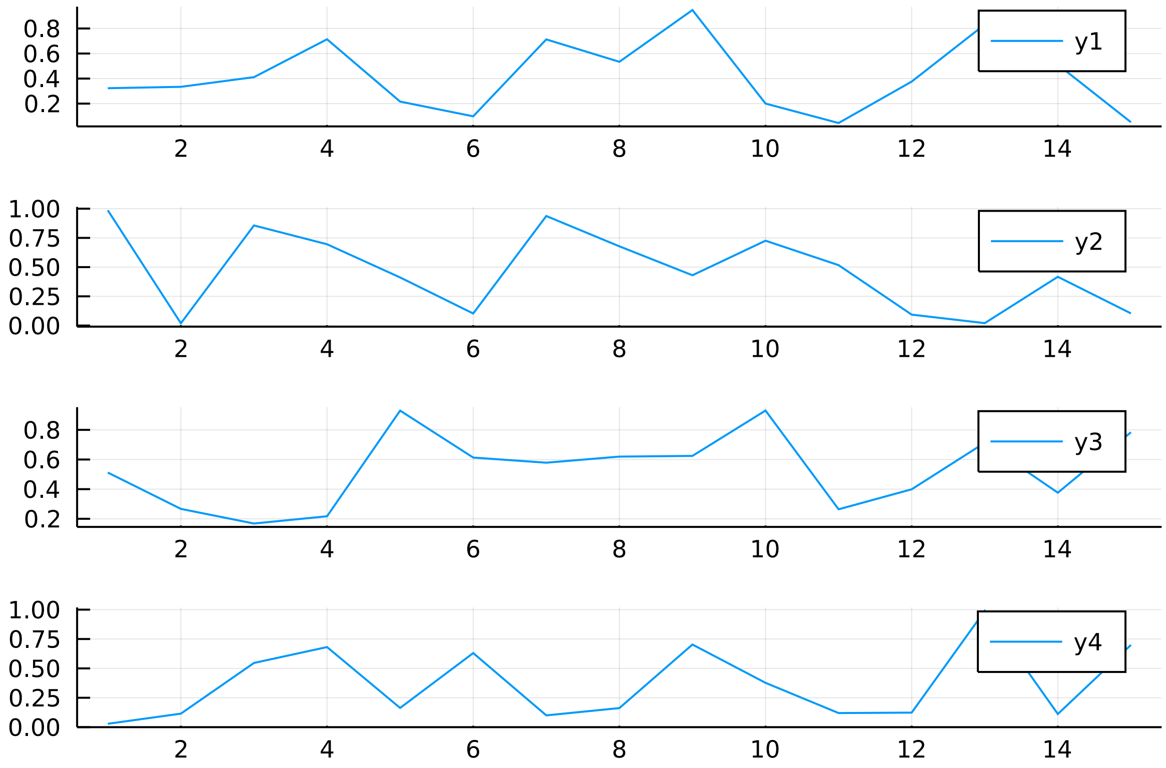
Out[53]:



In [54]:

```
y = rand(15,4)
plot(x,y,layout =(4,1))
```

Out[54]:



Data Visualization On Real Data

In [55]:

```
first(iris,5) # we are using iris dataset
```

Out[55]:

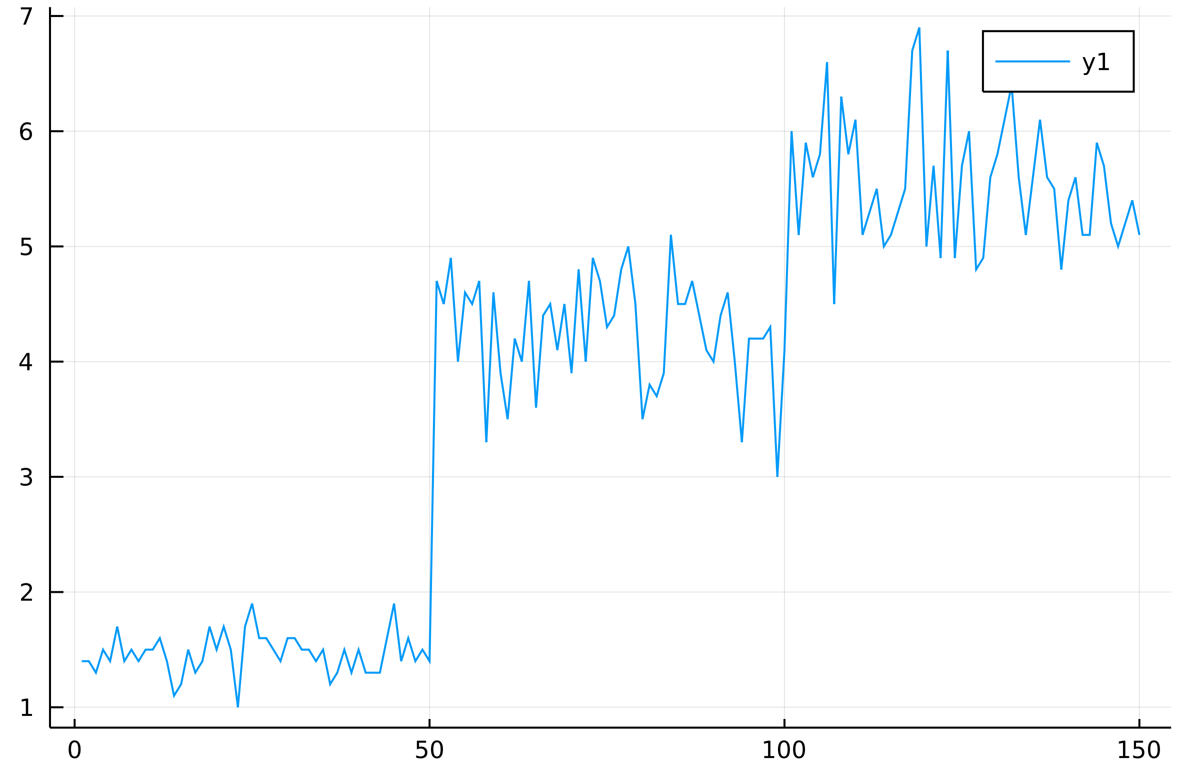
5 rows × 6 columns

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
	Int64	Float64	Float64	Float64	Float64	String
1	1	5.1	3.5	1.4	0.2	Iris-setosa
2	2	4.9	3.0	1.4	0.2	Iris-setosa
3	3	4.7	3.2	1.3	0.2	Iris-setosa
4	4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	5.0	3.6	1.4	0.2	Iris-setosa

In [56]:

```
plot(iris.PetalLengthCm)
```

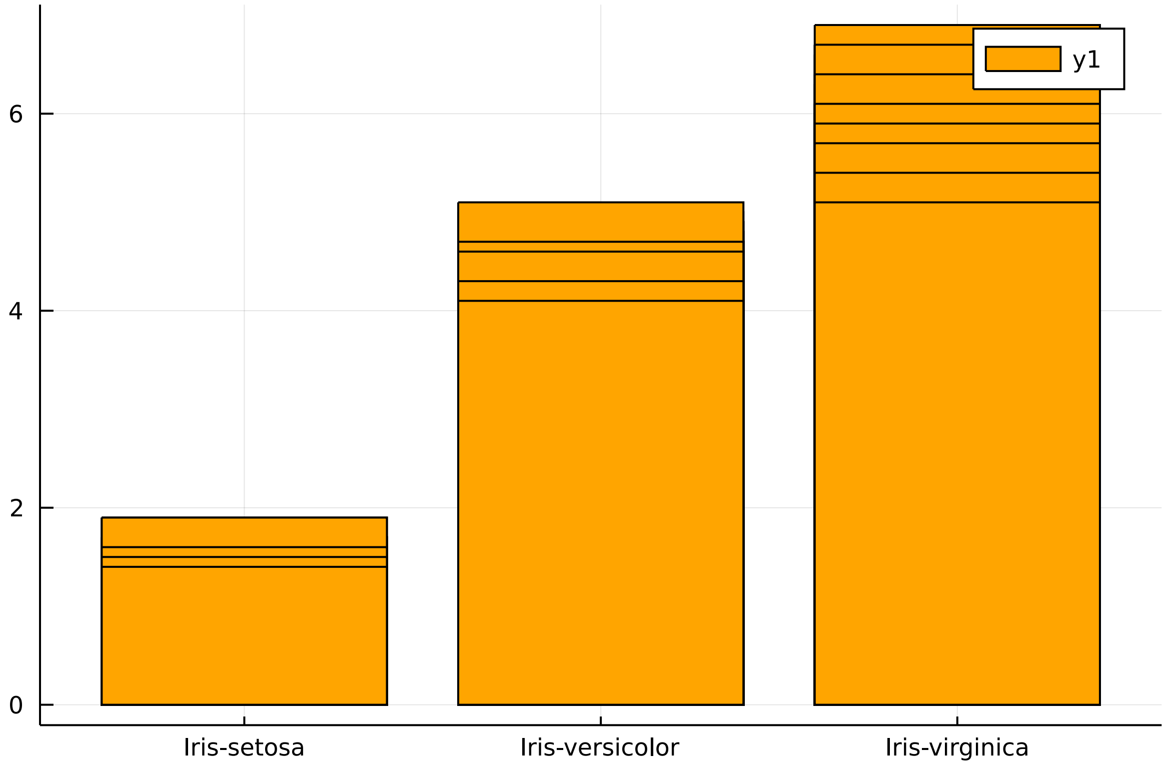
Out[56]:



In [57]:

```
bar(iris.Species, iris.PetalLengthCm , color="orange")
```

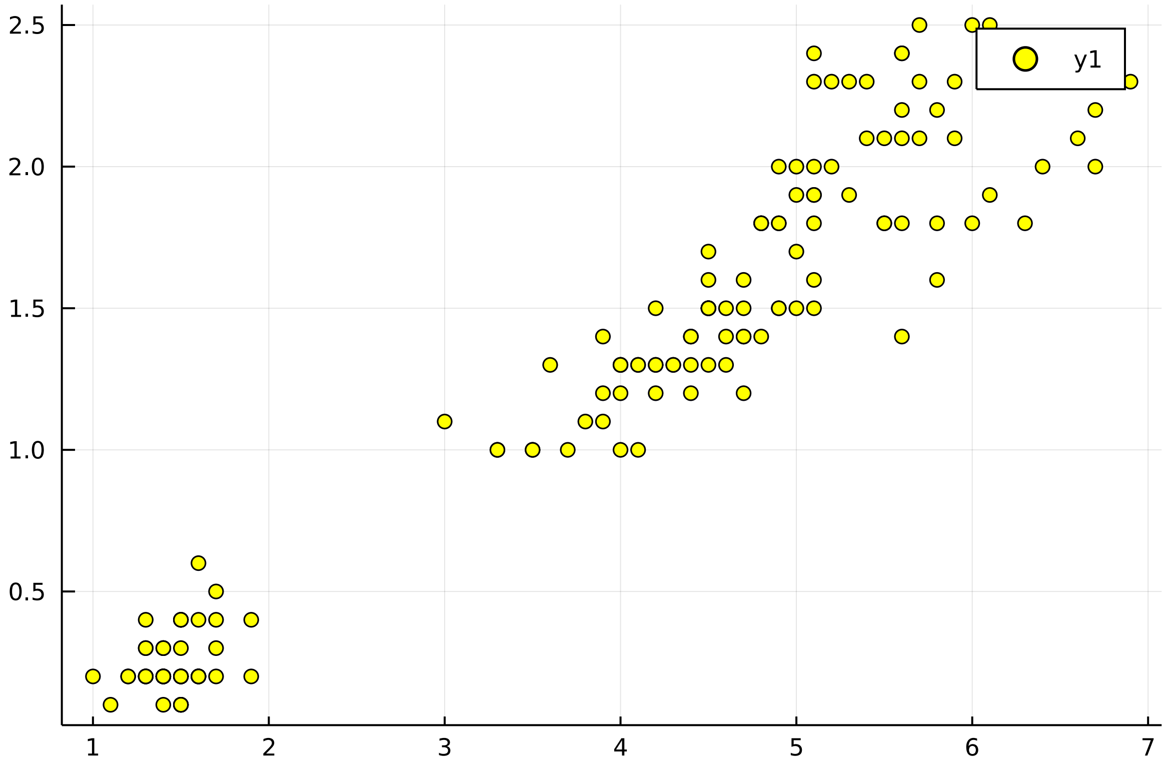
Out[57]:



In [58]:

```
scatter(iris.PetalLengthCm , iris.PetalWidthCm , markercolor = "yellow")
```

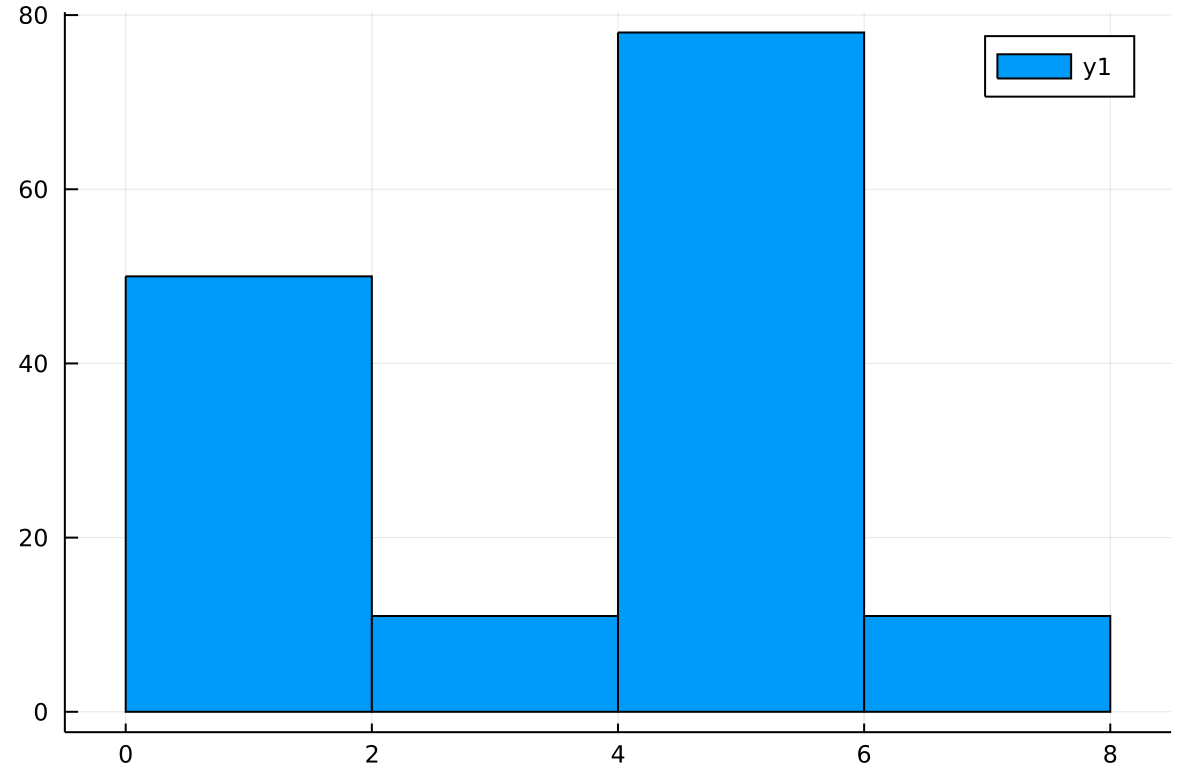
Out[58]:



In [59]:

```
histogram(iris.PetalLengthCm)
```

Out[59]:



Working With Database : SQL Lite

In [60]:

```
### downlaod package

# using Pkg
# Pkg.add("SQLite")
```

In [61]:

```
using SQLite
```

Operation with Database

- **DDL** : Create, Alter and Drop - *SQLite.DB*
- **DML** : Insert, Update and Delete - *SQLite.execute*
- **DQL** : Select - *SQLite.Query*

In [62]:

```
# create the database
db = SQLite.DB("Movies")
```

Out[62]:

```
SQLite.DB("Movies")
```

In [63]:

```
SQLite.execute(db , "CREATE TABLE IF NOT EXISTS movies(movie_id REAL,movie_name TEXT,locati
```

Out[63]:

```
101
```

In [64]:

```
# 101 means table create successfully...
```

In [65]:

```
SQLite.tables(db) # it shows how many tables present in the database.
```

Out[65]:

```
(name = ["movies"],)
```

In [66]:

```
# insert the record in table

SQLite.execute(db,"INSERT INTO movies(movie_id,movie_name,location) VALUES (1,'Avengers','U
```

Out[66]:

```
101
```

In [67]:

```
# check wheather table has data or not

using DataFrames
DataFrame(DBInterface.execute(db,"SELECT * FROM movies"))
```

Out[67]:

2 rows × 3 columns

	movie_id	movie_name	location
	Float64	String	String
1	1.0	Avengers	USA
2	1.0	Avengers	USA

In [68]:

```
# columns information of table

DataFrame(SQLite.columns(db,"movies"))
```

Out[68]:

3 rows × 6 columns

	cid	name	type	notnull	dflt_value	pk
	Int64	String	String	Int64	Missing	Int64
1	0	movie_id	REAL	0	*missing*	0
2	1	movie_name	TEXT	0	*missing*	0
3	2	location	TEXT	0	*missing*	0

In [69]:

```
chinook = SQLite.DB("chinook.db")
```

Out[69]:

```
SQLite.DB("chinook.db")
```

In [70]:

```
print(SQLite.tables(chinook))
```

```
(name = ["albums", "sqlite_sequence", "artists", "customers", "employees",
"genres", "invoices", "invoice_items", "media_types", "playlists", "playlist
_track", "tracks", "sqlite_stat1"],)
```

In [71]:

```
DataFrame(DBInterface.execute(chinook,"SELECT * FROM employees"))
```

Out[71]:

8 rows × 15 columns (omitted printing of 9 columns)

	EmployeeId	LastName	FirstName	Title	ReportsTo	BirthDate
	Int64	String	String	String	Int64?	String
1	1	Adams	Andrew	General Manager	*missing*	1962-02-18 00:00:00
2	2	Edwards	Nancy	Sales Manager	1	1958-12-08 00:00:00
3	3	Peacock	Jane	Sales Support Agent	2	1973-08-29 00:00:00
4	4	Park	Margaret	Sales Support Agent	2	1947-09-19 00:00:00
5	5	Johnson	Steve	Sales Support Agent	2	1965-03-03 00:00:00
6	6	Mitchell	Michael	IT Manager	1	1973-07-01 00:00:00
7	7	King	Robert	IT Staff	6	1970-05-29 00:00:00
8	8	Callahan	Laura	IT Staff	6	1968-01-09 00:00:00

Important Topics *(You Have to Do It By Yourself)*

- Where Condition
- Group By
- Order By
- Having
- Join

Refrence Links :

- SQL Lite : https://www.sqlitetutorial.net/ (https://www.sqlitetutorial.net/)
- SQLite Package in Julia : https://juliadatabases.org/SQLite.jl/stable/ (https://juliadatabases.org/SQLite.jl/stable/)

Working With Python Package

In [107]:

```
# download package PyCall

# using Pkg
# Pkg.add("PyCall")
```

In [73]:

```
using PyCall
```

In [74]:

```
np = pyimport("numpy")
```

Out[74]:

```
PyObject <module 'numpy' from 'C:\\Users\\Admin\\.julia\\conda\\3\\lib\\site
-packages\\numpy\\__init__.py'>
```

In [75]:

```
a1 = np.array([2,3,4,5,6])
```

Out[75]:

```
5-element Vector{Int64}:
 2
 3
 4
 5
 6
```

In [76]:

```
println(np.mean(a1))
println(np.std(a1))
```

```
4.0
1.4142135623730951
```

FOR MORE CHECKOUT GEEKS FOR GEEKS BLOG :
Working with Python Package in Julia (https://www.geeksforgeeks.org/how-to-import-python-packages-in-julia/#:~:text=Users%20can%20import%20arbitrary%20Python,the%20Julia%20environment%20with%20Pkg.)

Machine Learning In Julia

In [77]:

```
### add ML packages
### add CSV,DataFrames package if you don't have installed this package run below command

# # scikit learn
# using Pkg
# Pkg.add("ScikitLearn")

# # CSV
# using Pkg
# Pkg.add("CSV")

### DataFrames
# using Pkg
# Pkg.add("DataFrames")
```

In [78]:

```
using ScikitLearn , CSV
```

```
┌ Info: Precompiling ScikitLearn [3646fa90-6ef7-5e7e-9f22-8aca16db6324]
└ @ Base loading.jl:1317
┌ Warning: Module StatsBase with build ID 115569723039701 is missing from th
e cache.
│ This may mean StatsBase [2913bbd2-ae8a-5f71-8c99-4fb6c76f3a91] does not su
pport precompilation but is imported by a module that does.
└ @ Base loading.jl:1008
┌ Info: Skipping precompilation since __precompile__(false). Importing Sciki
tLearn [3646fa90-6ef7-5e7e-9f22-8aca16db6324].
└ @ Base loading.jl:1025
```

In [79]:

```
using DataFrames
```

In [80]:

```
iris = CSV.read("Iris.csv",DataFrame)
```

Out[80]:

150 rows × 6 columns

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
	Int64	Float64	Float64	Float64	Float64	String
1	1	5.1	3.5	1.4	0.2	Iris-setosa
2	2	4.9	3.0	1.4	0.2	Iris-setosa
3	3	4.7	3.2	1.3	0.2	Iris-setosa
4	4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	5.0	3.6	1.4	0.2	Iris-setosa
6	6	5.4	3.9	1.7	0.4	Iris-setosa
7	7	4.6	3.4	1.4	0.3	Iris-setosa
8	8	5.0	3.4	1.5	0.2	Iris-setosa
9	9	4.4	2.9	1.4	0.2	Iris-setosa
10	10	4.9	3.1	1.5	0.1	Iris-setosa
11	11	5.4	3.7	1.5	0.2	Iris-setosa
12	12	4.8	3.4	1.6	0.2	Iris-setosa
13	13	4.8	3.0	1.4	0.1	Iris-setosa
14	14	4.3	3.0	1.1	0.1	Iris-setosa
15	15	5.8	4.0	1.2	0.2	Iris-setosa
16	16	5.7	4.4	1.5	0.4	Iris-setosa
17	17	5.4	3.9	1.3	0.4	Iris-setosa
18	18	5.1	3.5	1.4	0.3	Iris-setosa
19	19	5.7	3.8	1.7	0.3	Iris-setosa
20	20	5.1	3.8	1.5	0.3	Iris-setosa
21	21	5.4	3.4	1.7	0.2	Iris-setosa
22	22	5.1	3.7	1.5	0.4	Iris-setosa
23	23	4.6	3.6	1.0	0.2	Iris-setosa
24	24	5.1	3.3	1.7	0.5	Iris-setosa
25	25	4.8	3.4	1.9	0.2	Iris-setosa
26	26	5.0	3.0	1.6	0.2	Iris-setosa
27	27	5.0	3.4	1.6	0.4	Iris-setosa
28	28	5.2	3.5	1.5	0.2	Iris-setosa
29	29	5.2	3.4	1.4	0.2	Iris-setosa
30	30	4.7	3.2	1.6	0.2	Iris-setosa
⋮	⋮	⋮	⋮	⋮	⋮	⋮

In [81]:

```
features = iris[:,[2,3,4,5]];
```

In [82]:

```
target = iris.Species;
```

In [83]:

```
features = Matrix(features)
```

Out[83]:

```
150×4 Matrix{Float64}:
 5.1  3.5  1.4  0.2
 4.9  3.0  1.4  0.2
 4.7  3.2  1.3  0.2
 4.6  3.1  1.5  0.2
 5.0  3.6  1.4  0.2
 5.4  3.9  1.7  0.4
 4.6  3.4  1.4  0.3
 5.0  3.4  1.5  0.2
 4.4  2.9  1.4  0.2
 4.9  3.1  1.5  0.1
 5.4  3.7  1.5  0.2
 4.8  3.4  1.6  0.2
 4.8  3.0  1.4  0.1
 ⋮
 6.0  3.0  4.8  1.8
 6.9  3.1  5.4  2.1
 6.7  3.1  5.6  2.4
 6.9  3.1  5.1  2.3
 5.8  2.7  5.1  1.9
 6.8  3.2  5.9  2.3
 6.7  3.3  5.7  2.5
 6.7  3.0  5.2  2.3
 6.3  2.5  5.0  1.9
 6.5  3.0  5.2  2.0
 6.2  3.4  5.4  2.3
 5.9  3.0  5.1  1.8
```

In [84]:

```
target = convert(Array,target)
```

Out[84]:

```
150-element Vector{String}:
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 "Iris-setosa"
 ⋮
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
 "Iris-virginica"
```

Logistic Regression

In [85]:

```
@sk_import linear_model: LogisticRegression
```

Out[85]:

```
PyObject <class 'sklearn.linear_model._logistic.LogisticRegression'>
```

In [86]:

```
log_reg_model = LogisticRegression()
```

Out[86]:

```
PyObject LogisticRegression()
```

In [87]:

```
fit!(log_reg_model,features,target)
```

Out[87]:

```
PyObject LogisticRegression()
```

In [88]:

```
predictions = predict(log_reg_model,features);
```

In [89]:

```
@sk_import metrics:accuracy_score
```

Out[89]:

```
PyObject <function accuracy_score at 0x0000000099E010D0>
```

In [90]:

```
accuracy_score(predictions , target)
```

Out[90]:

```
0.9733333333333334
```

Decision Tree

In [91]:

```
@sk_import tree:DecisionTreeClassifier
```

Out[91]:

```
PyObject <class 'sklearn.tree._classes.DecisionTreeClassifier'>
```

In [92]:

```
tree_model = DecisionTreeClassifier()
```

Out[92]:

```
PyObject DecisionTreeClassifier()
```

In [93]:

```
fit!(tree_model,features,target)
```

Out[93]:

```
PyObject DecisionTreeClassifier()
```

In [94]:

```
predictions = predict(tree_model,features);
```

In [95]:

```
accuracy_score(predictions , target)
```

Out[95]:

```
1.0
```

Random Forest

In [96]:

```
@sk_import ensemble: RandomForestClassifier
```

Out[96]:

```
PyObject <class 'sklearn.ensemble._forest.RandomForestClassifier'>
```

In [97]:

```
random_forest = RandomForestClassifier(n_estimators = 5)
```

Out[97]:

```
PyObject RandomForestClassifier(n_estimators=5)
```

In [98]:

```
fit!(random_forest , features , target)
```

Out[98]:

```
PyObject RandomForestClassifier(n_estimators=5)
```

In [99]:

```
predictions = predict(random_forest , features);
```

In [100]:

```
accuracy_score(predictions , target)
```

Out[100]:

```
0.9866666666666667
```

Train Test Split

In [101]:

```
@sk_import model_selection : train_test_split
```

```
┌ Warning: Module model_selection has been ported to Julia - try `import Sci
kitLearn: CrossValidation` instead
└ @ ScikitLearn.Skcore C:\Users\Admin\.julia\packages\ScikitLearn\NJwUf\src
\Skcore.jl:179
```

Out[101]:

```
PyObject <function train_test_split at 0x0000000099DD7280>
```

In [102]:

```
x_train , x_test , y_train , y_test = train_test_split(features , target , test_size = 0.2
```

In [103]:

```
log_reg_2 = LogisticRegression(solver="lbfgs", max_iter=1000)

# add hyperparameters to ignore the warning.
```

Out[103]:

```
PyObject LogisticRegression(max_iter=1000)
```

In [104]:

```
fit!(log_reg_2,x_train,y_train)
```

Out[104]:

```
PyObject LogisticRegression(max_iter=1000)
```

In [105]:

```
preditions = predict(log_reg_2 , x_test);
```

In [106]:

```
accuracy_score(preditions , y_test)
```

Out[106]:

```
1.0
```

THANK YOU 🎉 🎊

For More Details Do Check Out Julia Official Documentation.
https://docs.julialang.org/en/v1/ (https://docs.julialang.org/en/v1/)