Lecture_Visualization

August 2, 2023

1 Data Visualization

In this lecture/tutorial we will work together on different plotting exercises to be able to represent our data in the best possible manner. Let's start with first generationg some data into our julia environment. For this we will be using the package *DataSci4Chem.jl*.

1.1 Data Generation

- Generate a vector of values between 1 and 3 (X) with a size of 10x1.
- Generate a vector of Y by first calculating the sin(X) and then add some noise to those values.

[ ]:
```
using DataSci4Chem

X = collect(range(1,3,length=10))  # Generate some x values
Y = sin.(X) .+ rand(10,1)          # Generate some y values
println(X)
println(Y)
```

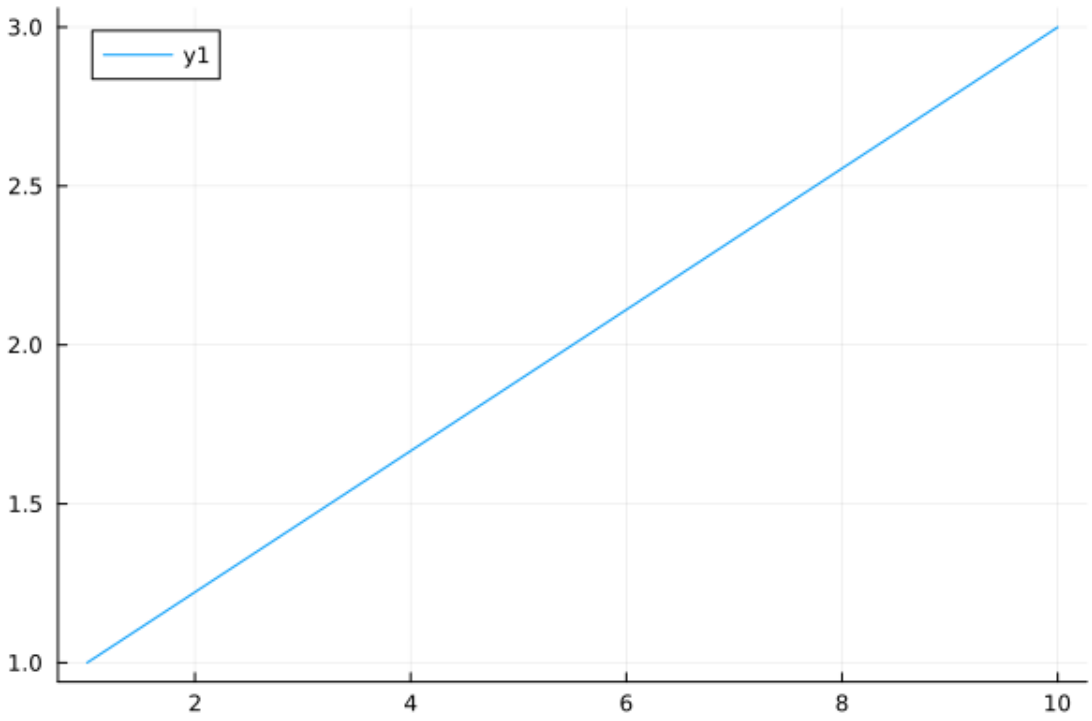
```
[1.0, 1.2222222222222223, 1.4444444444444444, 1.6666666666666667,
1.8888888888888888, 2.111111111111111, 2.3333333333333335, 2.5555555555555554,
2.7777777777777777, 3.0]
[0.8712137033336134; 1.5986274515148053; 1.7179799246416905; 1.2247578188352954;
1.4762307294330193; 1.1725526046867838; 1.027146189864447; 0.9232128675182156;
0.6531703967786979; 0.30569793689951974]
```

1.2 E1:

Let's generate a line plot of the variable X and answer the below questions.

1. what is plotted in the x axis?
2. what is plotted in the y axis?
3. what is actually being plotted?

[ ]:
```
plot(X)
```



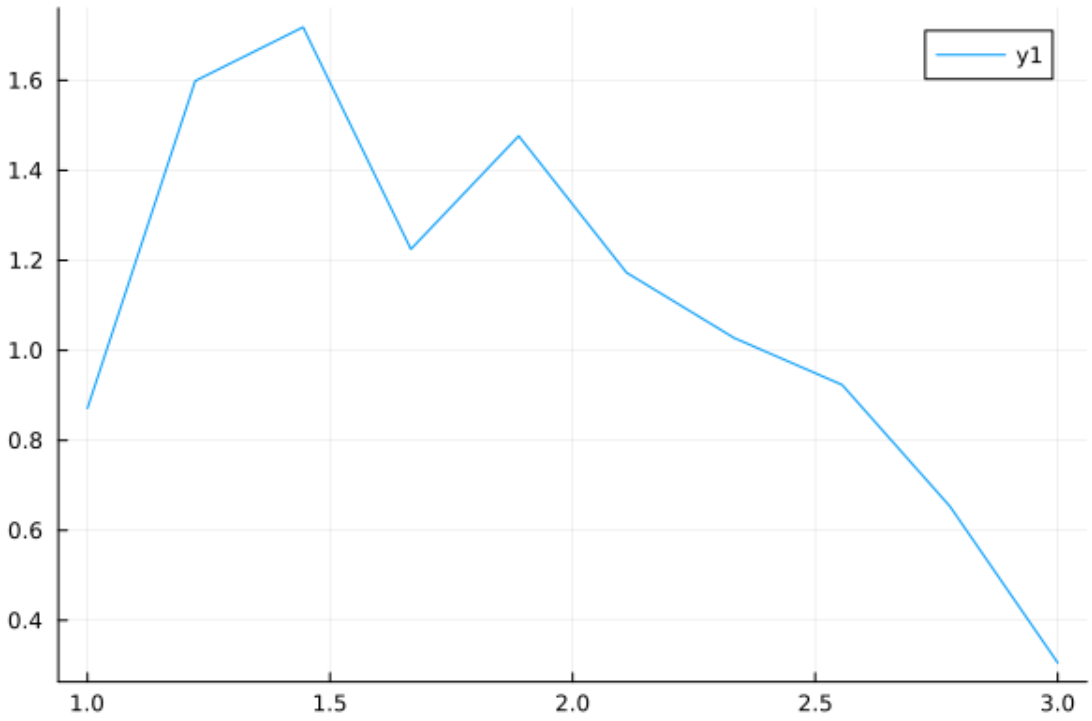
Answers to E1

1. The x axis is the index of individual values in the X vector.
2. The actual values of X are plotted in in the y domain.
3. plot(1:length(X),X)

1.3 E2:

Let's now plot the X vs Y.

[ ]:
```
plot(X,Y)
```



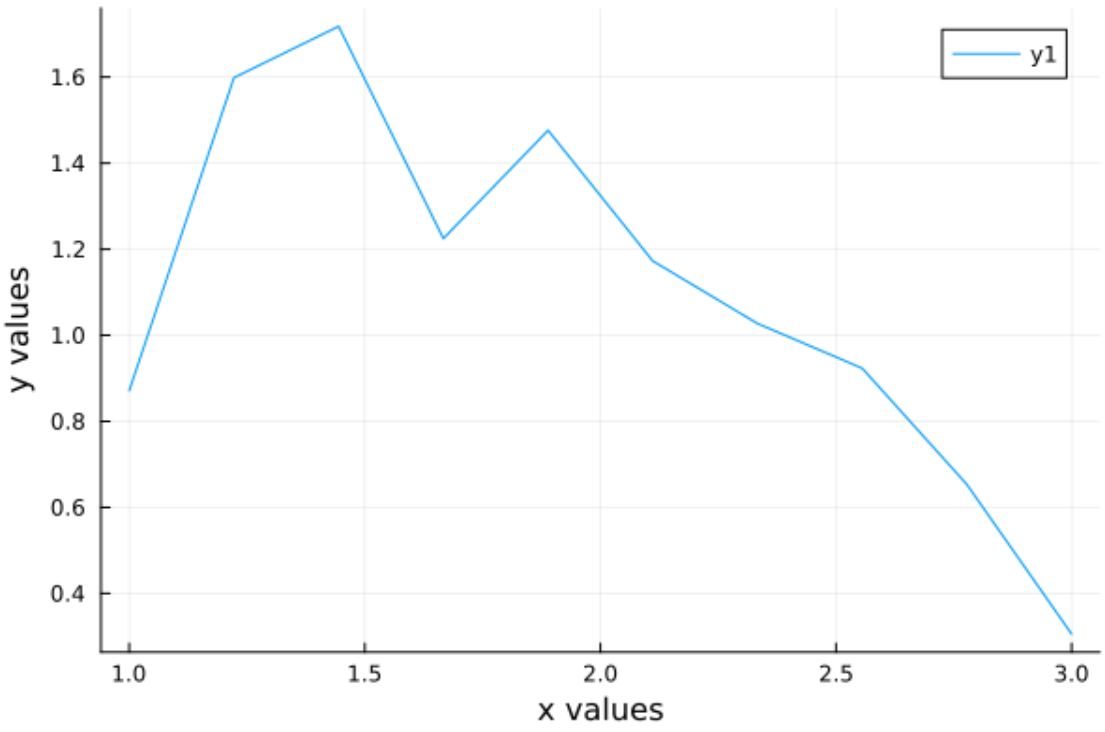
1.4 E3:

Let's try to set the x and y label in this plot. How can we do that?

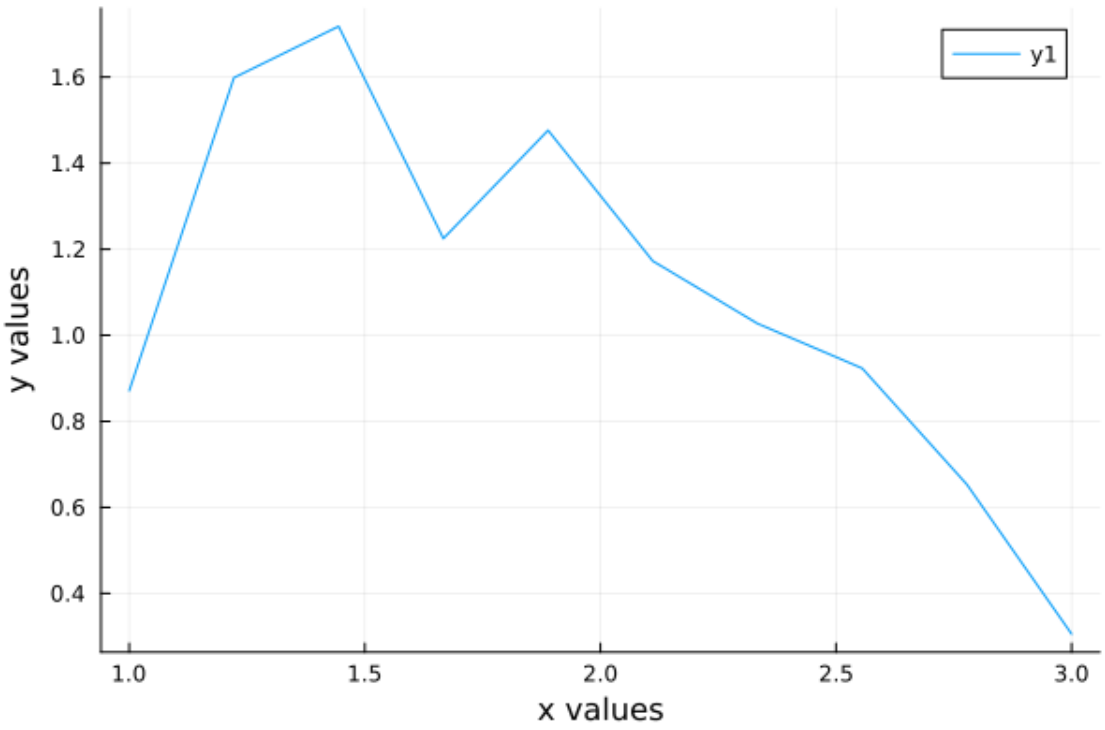
Answers to E3

For that there are two different ways of of seeting this up. The first way is by specifying the labels as an attribute in the plot(-) the second way is to use the function xlabel!(-).

[ ]:
```
plot(X,Y,xlabel= "x values",ylabel = "y values")
```



[ ]:
```
# or
plot(X,Y)
xlabel!("x values")
ylabel!("y values")
```



1.5 E4:

What is the type of values fed to xlabel?

[ ]:
```
typeof("x values")
```

```
String
```

Answers to E4

The type of value fed to these parameters is typically string, hence the quotation marks around the values.

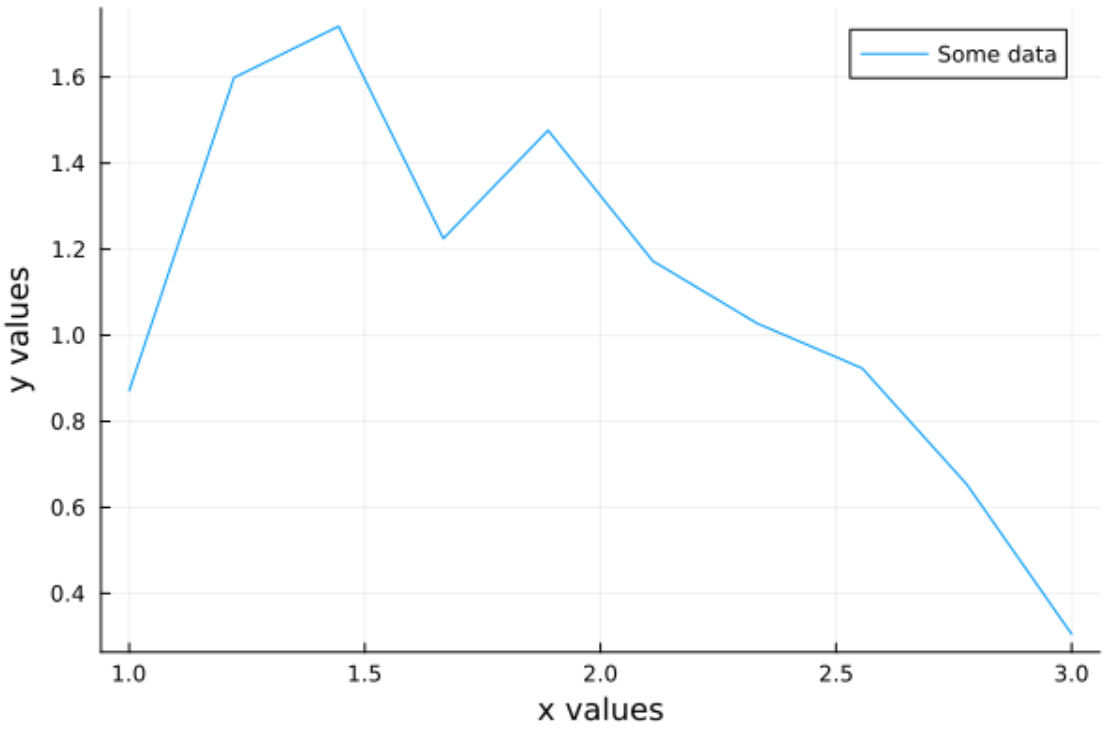
1.6 E5:

How can we adjust the legend in our plot?

Answers to E5

The legend in the plot can be adjusted by providing that parameter to the *plot(-)* function.

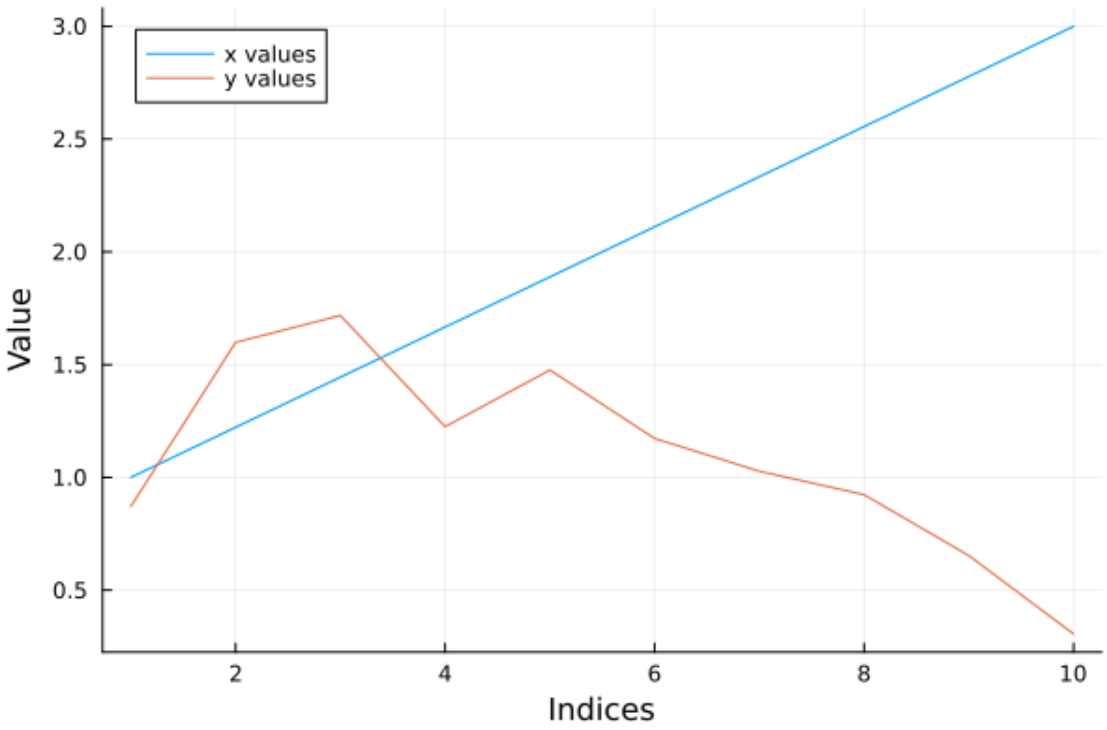
[ ]:
```
plot(X,Y,label = "Some data")
xlabel!("x values")
ylabel!("y values")
```



1.7 E6:

How can we add a second data series to your plot? Plot the individual X and Y vectors as two separate lines.

[ ]:
```
plot(X,label = "x values")
plot!(Y,label = "y values")
xlabel!("Indices")
ylabel!("Value")
```



1.8 E7:

How can we control the color of the lines? Let's have the X plotted in black and Y plotted in orange.

Answers to E7

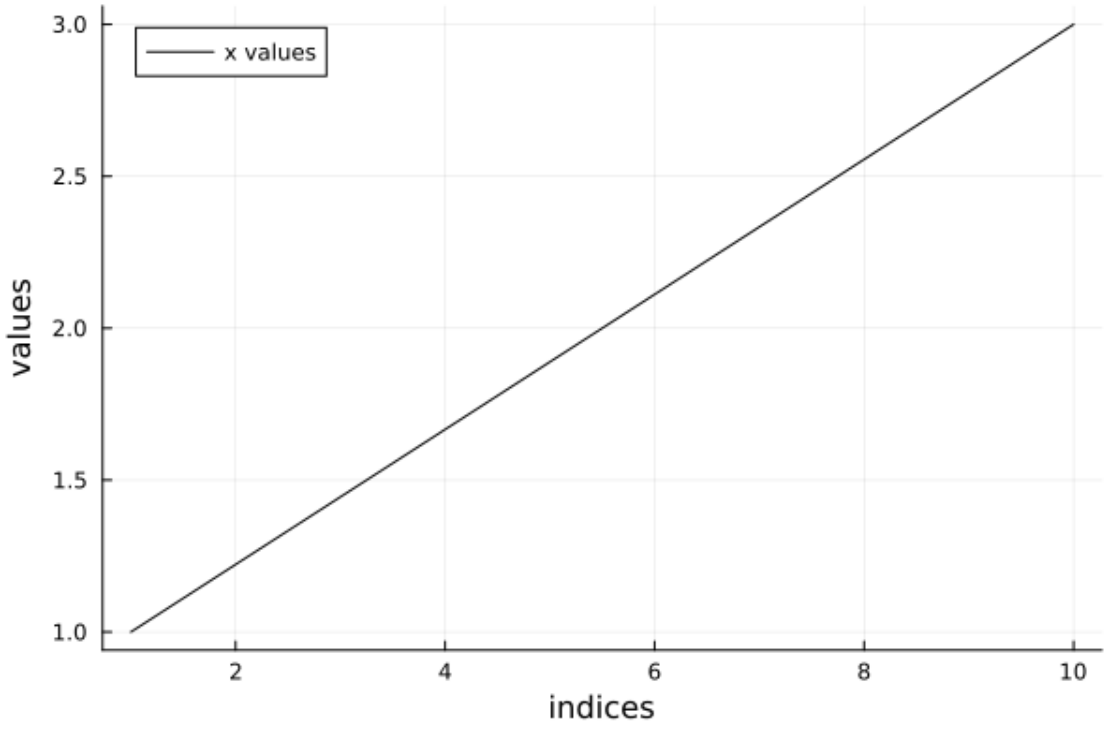
The color of the lines is set through the parameter "lc" within the *plot(-)* function. Please note that when specifying the colors you need to set them as ":black" for julia to recognize them as specific settings.

[ ]:
```
# Wrong color setting

plot(X,
    label = "some data",
    lc = black,
    xlabel = "x values",
    ylabel = "y values")
```

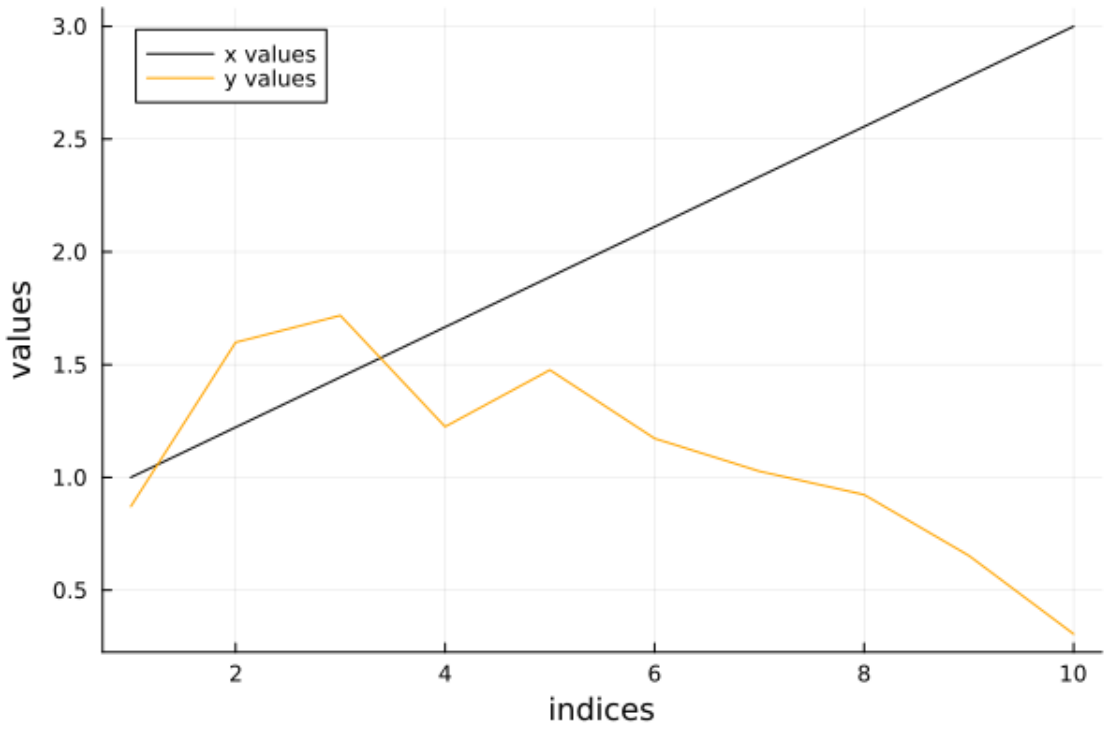
```
UndefVarError: black not defined



Stacktrace:
```

```
[1] top-level scope

  @ ~/Desktop/dev/pkgs/DataSci4Chem.jl/Notebooks/Lecture_Visualization.ipynb:3
```

[ ]:
```
# Correct color setting

plot(X,
    label = "x values",
    lc = :black,
    xlabel = "indices",
    ylabel = "values")
```



[ ]:
```
plot!(Y,
    label = "y values",
    lc = :orange,
    xlabel = "indices",
    ylabel = "values")
```



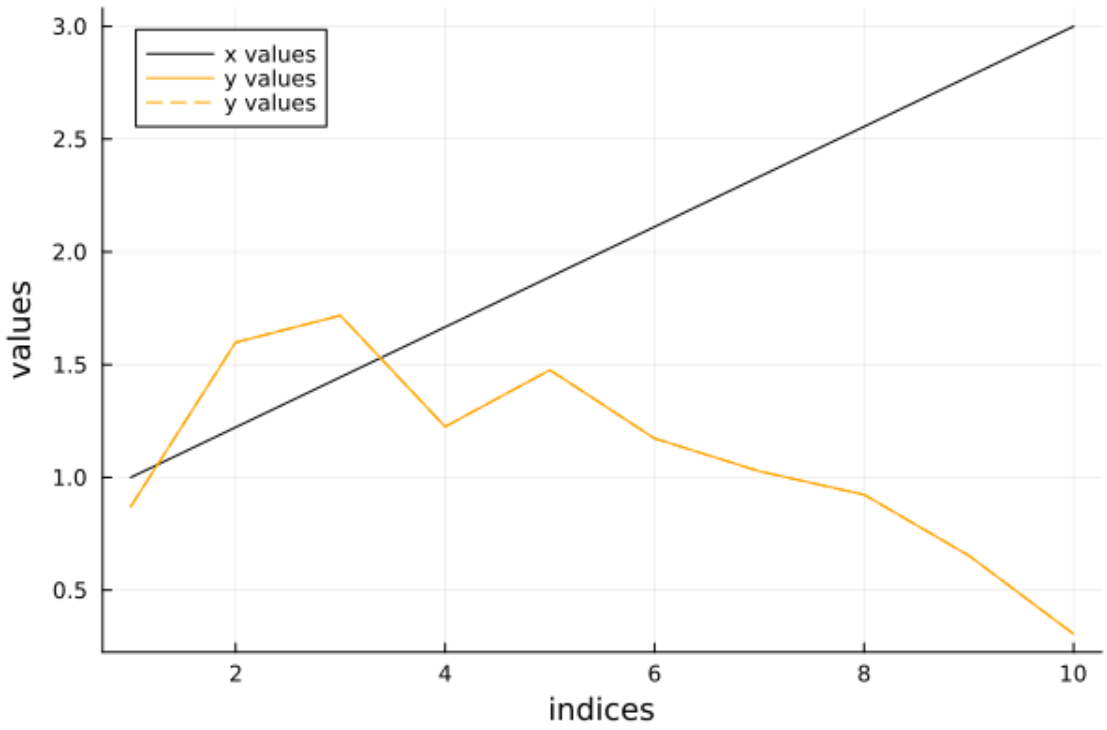
1.9 E9:

How about the line style? What if we want the lines to have different styles. Let's set the Y line style to dash lines.

Answers to E9

The line style can be set using the attribute "ls". In this case also you need to use the same format for the line color (i.e. ":dash" for dashed line).

[ ]:
```
plot!(Y,
    label = "y values",
    lc = :orange,
    xlabel = "indices",
    ylabel = "values",
    ls = :dash)
```



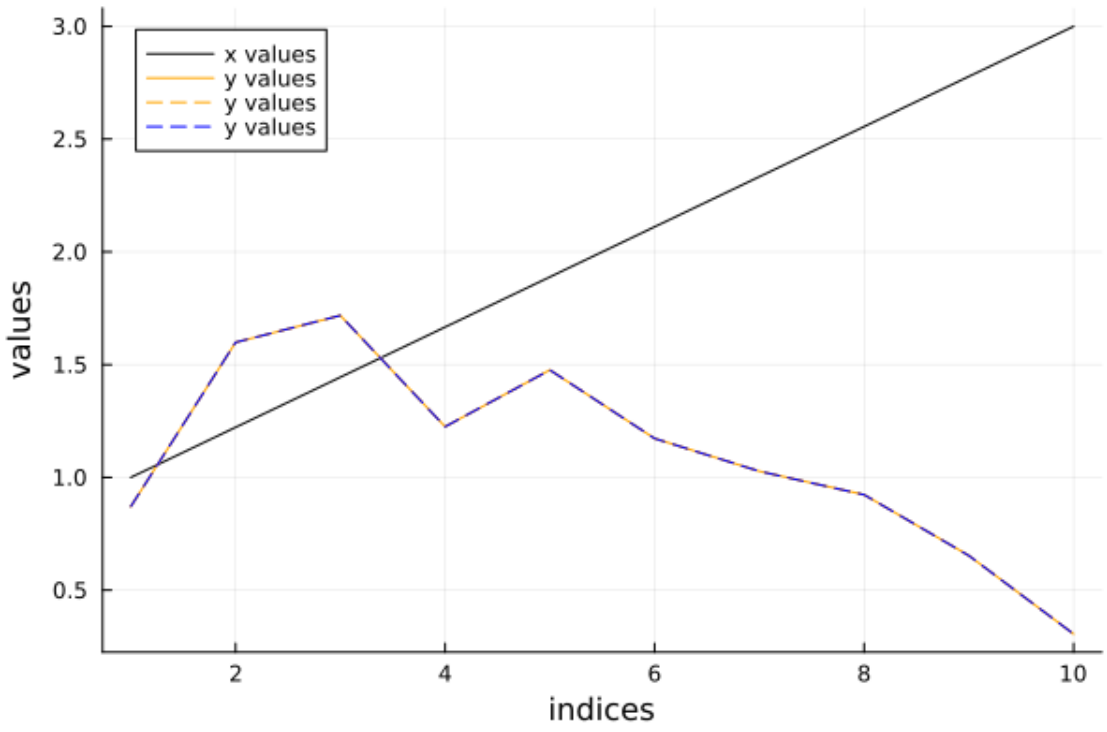
1.10 E10:

It seems that nothing has changed. Why?

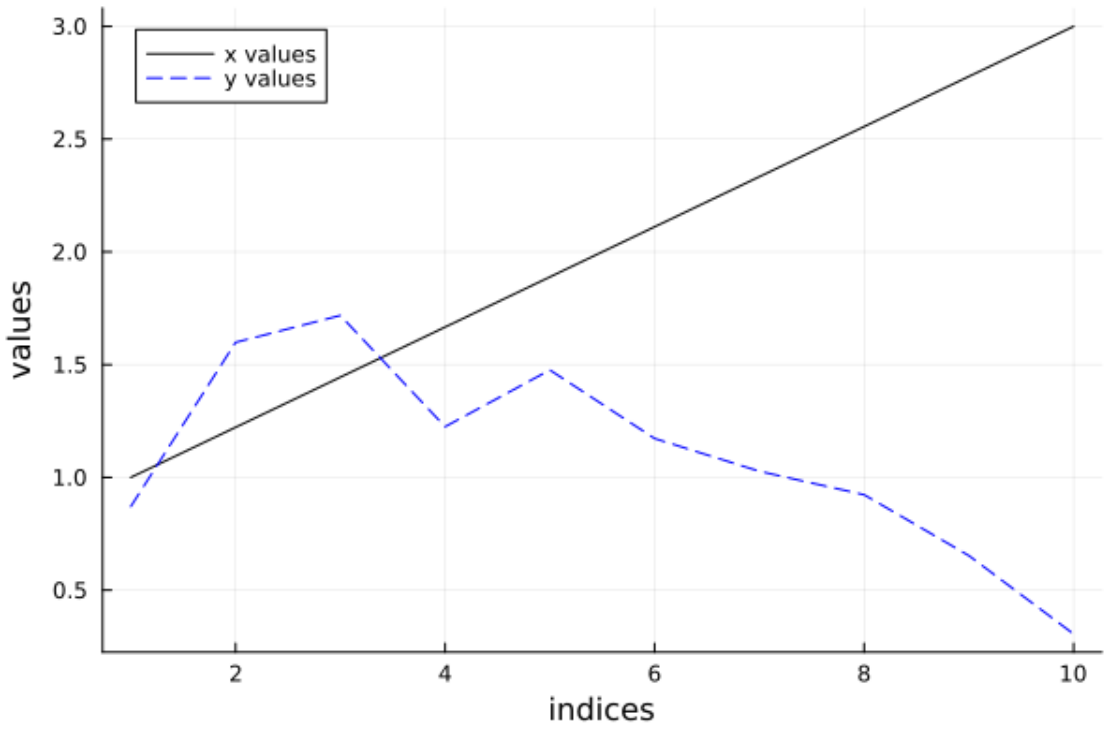
Answers to E10

Julia is using currently stored figure in the memory to overlay the new plot on top. This implies that the orang dots are sitting on top of the orange line, thus not visible (please look at the legend). To change this you can either change also the color of the dots or restart the plot from scratch.

[ ]:
```
plot!(Y,
    label = "y values",
    lc = :blue,
    xlabel = "indices",
    ylabel = "values",
    ls = :dash)
```



[ ]:
```
plot(X,
    label = "x values",
    lc = :black)

plot!(Y,
    label = "y values",
    lc = :blue,
    xlabel = "indices",
    ylabel = "values",
    ls = :dash)
```



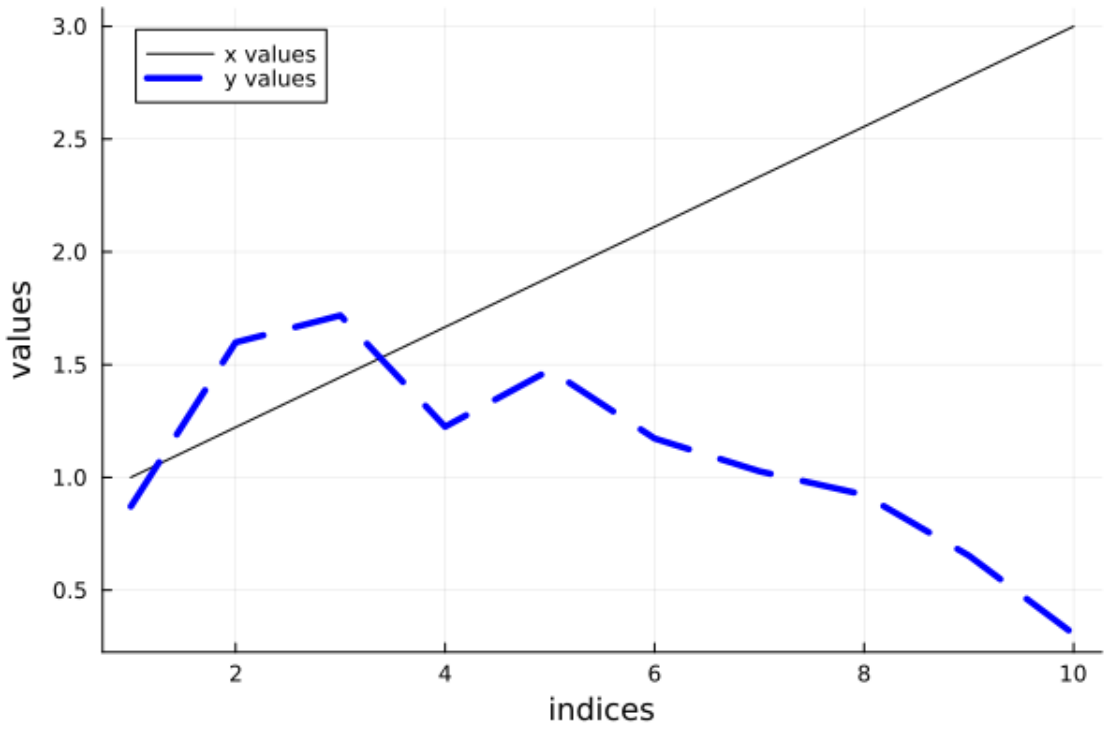
1.11 E11:

What about the line thickness? Let's try to increase the Y line thickness.

Answers to E11

You can set the line thickness with a parameter called "lw" (i.e. line width) directly provided to the function *plot(-)*.

[ ]:
```
plot(X,
    label = "x values",
    lc = :black)

plot!(Y,
    label = "y values",
    lc = :blue,
    xlabel = "indices",
    ylabel = "values",
    ls = :dash,
    lw = 3.5)
```



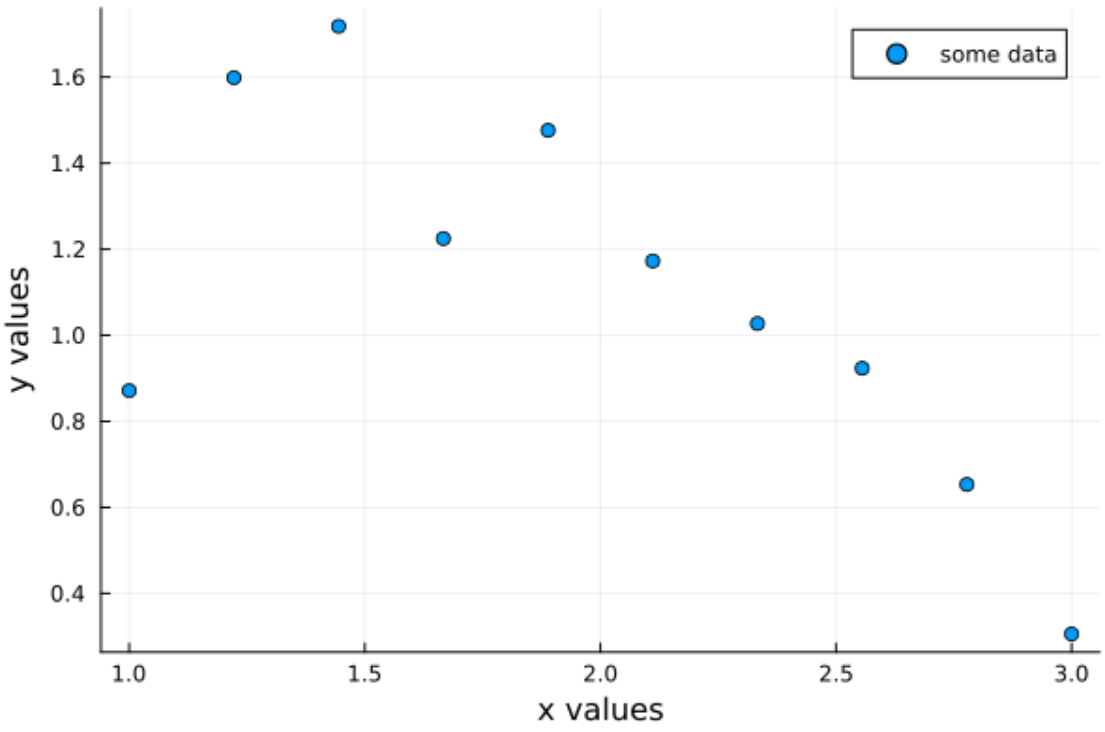
1.12 E12:

What if we want to have a scatter plot of X vs Y.

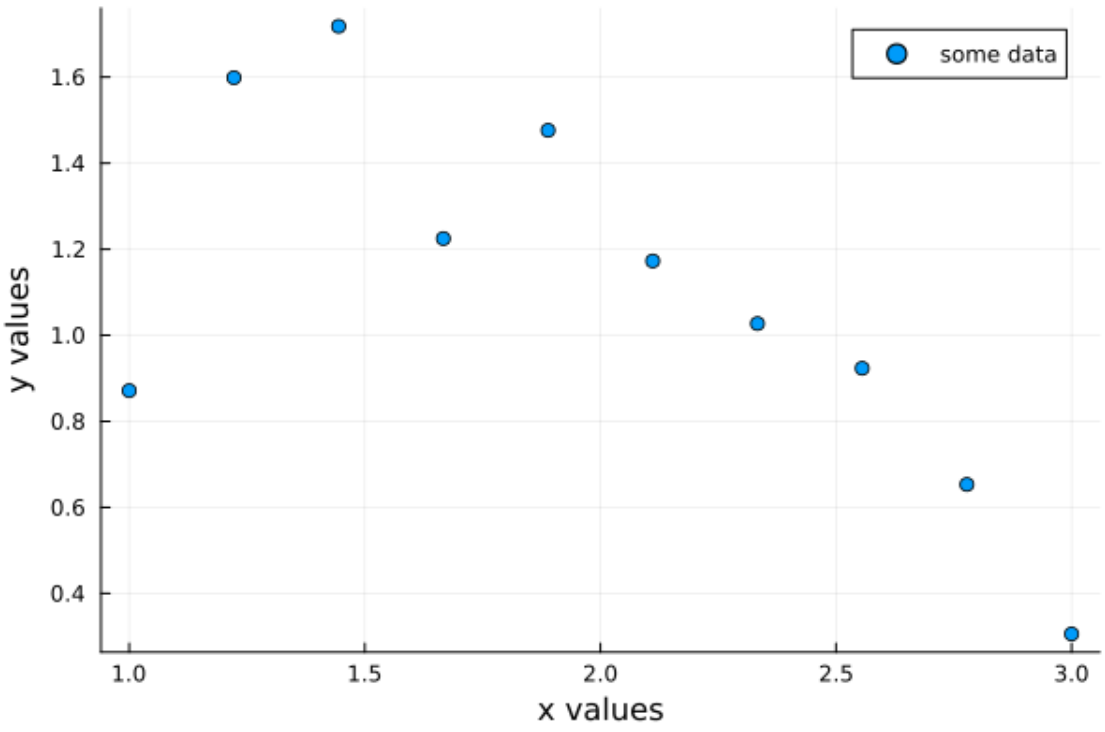
Answers to E12

This can be achieved either by using the function *scatter(-)* or defining the plot type via "st" in the function *plot(-)*.

[ ]:
```
scatter(X,Y,
    label = "some data",
    xlabel = "x values",
    ylabel = "y values")
```



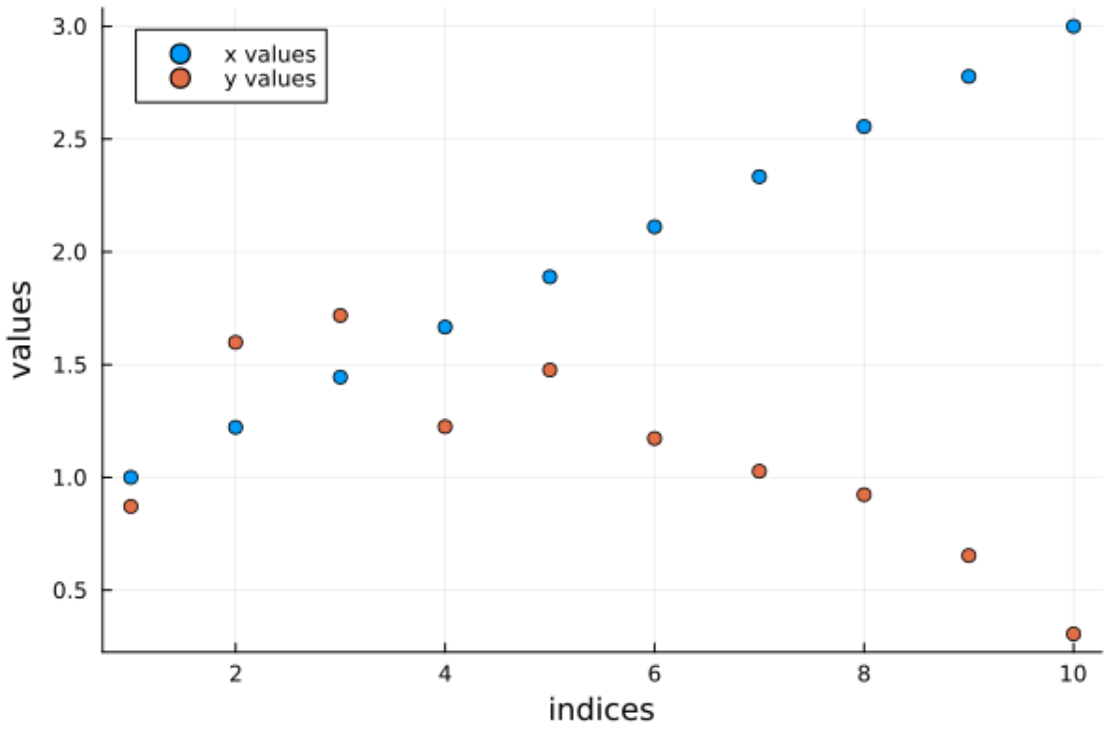
[ ]:
```
plot(X,Y,
    label = "some data",
    st = :scatter,
    xlabel = "x values",
    ylabel = "y values")
```



1.13 E13:

Let's try to plot X and Y as separate scatter plots.

[ ]:
```
scatter(X,
    label = "x values",
    xlabel = "indices",
    ylabel = "values")

scatter!(Y,
    label = "y values")
```



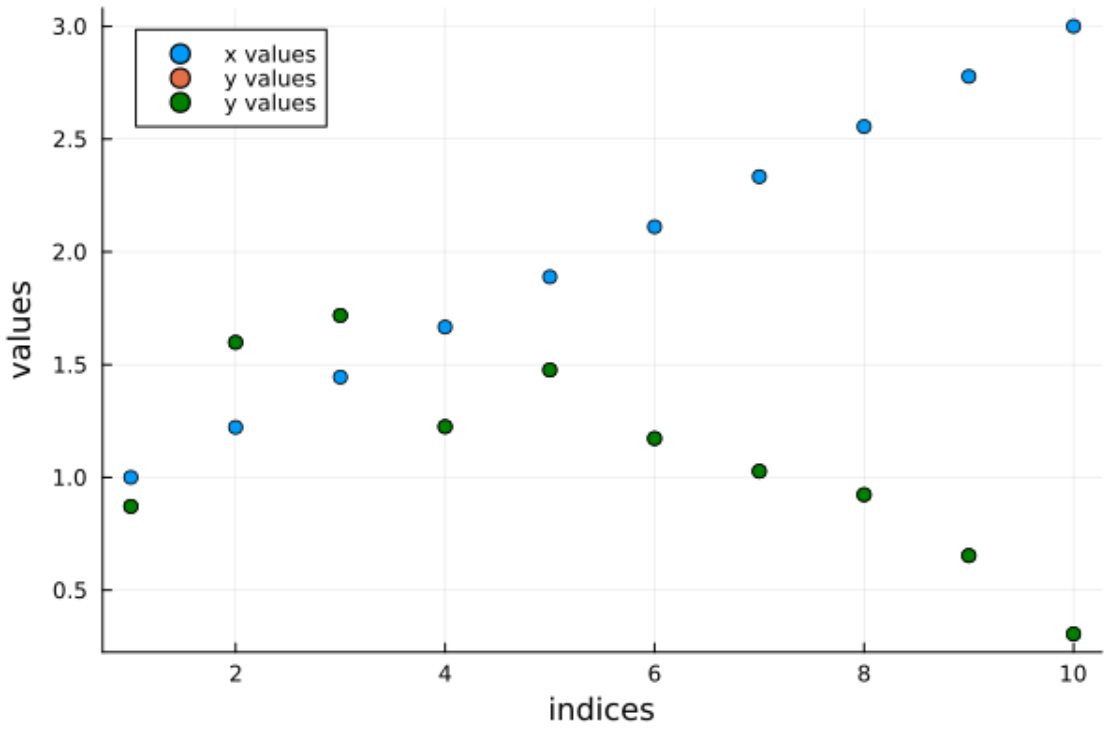
1.14 E14:

I really do not like the color of Y. Can we make it green?

Answers to E14

The color of markers can be adjusted using a parameter called "mc" (i.e. marker color). Please note that we keep the same formatting for this parameter as the line color.

[ ]:
```
scatter!(Y,
    label = "y values",
    mc = :green)
```



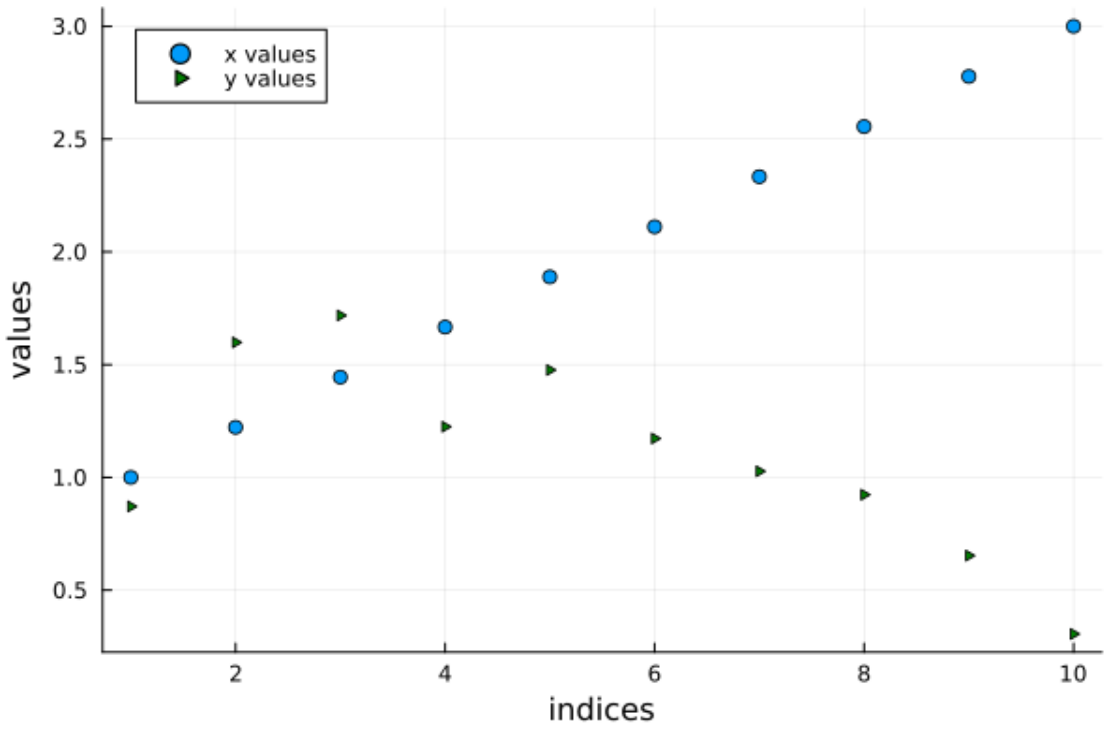
1.15 E15:

But now I am not able to distinguish the two series from each other. Is it possible to change the shape of the X?

Answers to E15

To set the shape of the markers in scatter plots, you need to adjust the parameter "shape" in the *plot(-)* function.

[ ]:
```
scatter(X,
    label = "x values",
    xlabel = "indices",
    ylabel = "values")

scatter!(Y,
    label = "y values",
    mc = :green,
    shape = :>)
```



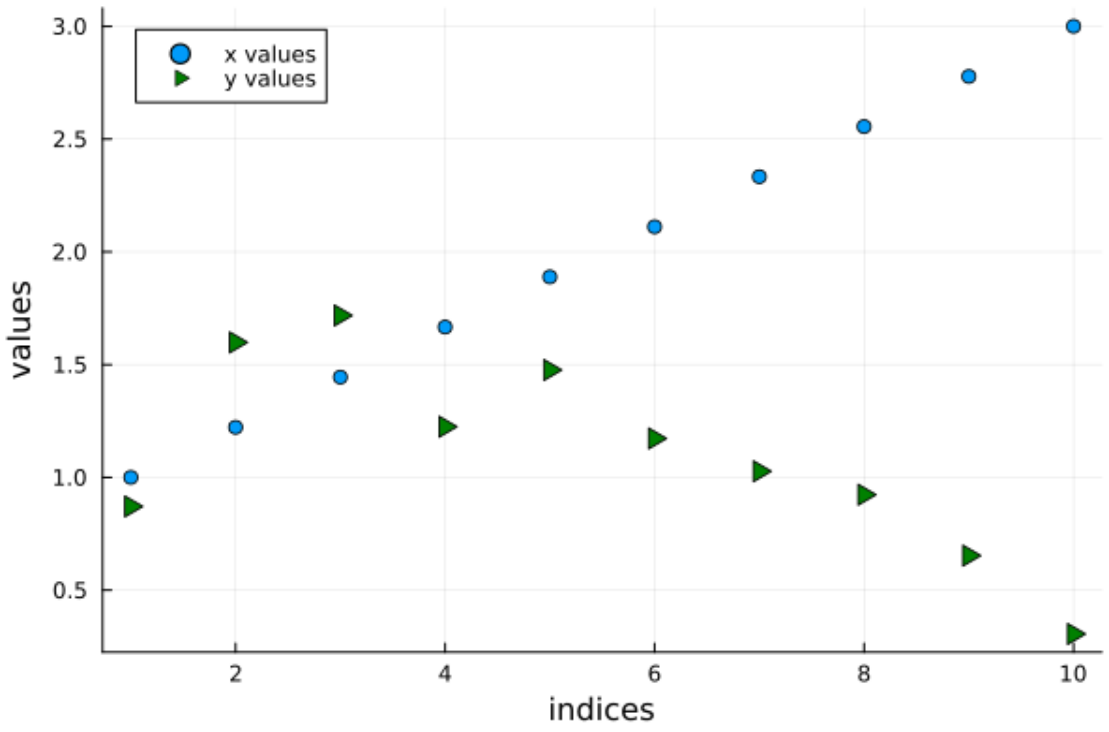
1.16 E16:

Now the marker of Y is too small. I am old and not able to see it well. Can you increase the size of that?

Answers to E16

For the marker size you can use the parameter "ms" in the *plot(-)* function.

[ ]:
```
scatter(X,
    label = "x values",
    xlabel = "indices",
    ylabel = "values")

scatter!(Y,
    label = "y values",
    mc = :green,
    shape = :>,
    ms = 8)
```

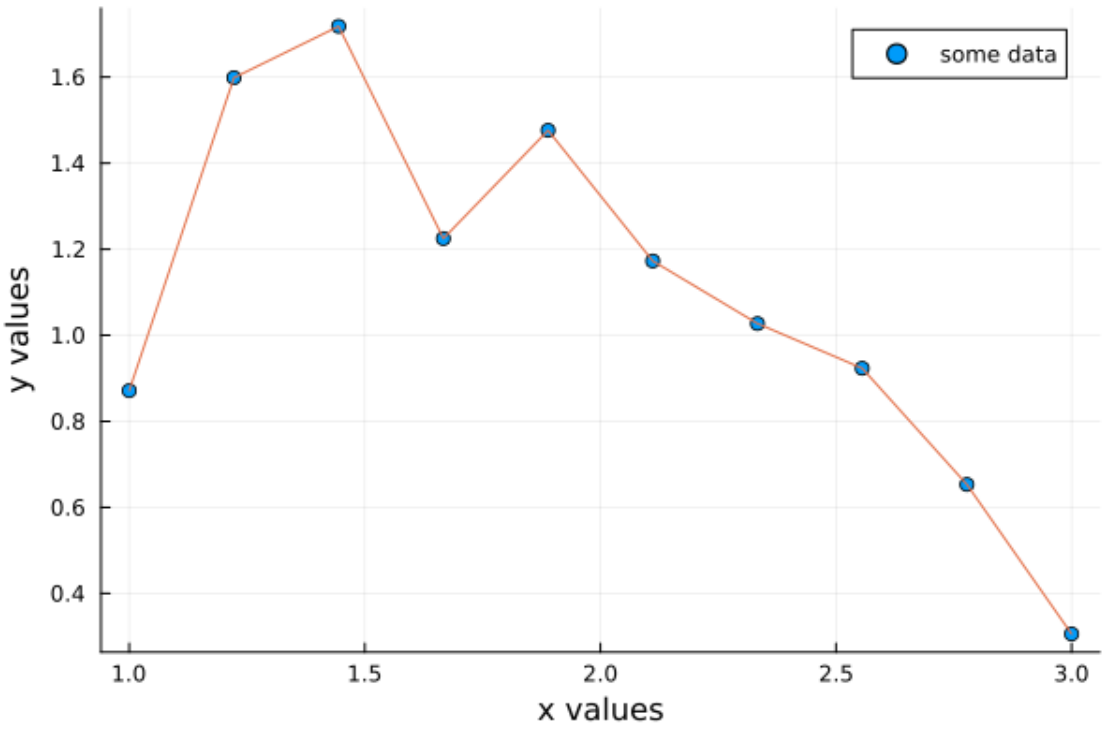


So far we have put a single type of plots into a frame.

1.17 E17:

Can we combine multiple plot types together? Let's try to plot X vs Y first as a scatter plot and then as a line plot in a single frame (i.e. the same figure).

[ ]:
```
scatter(X,Y,
    label = "some data",
    xlabel = "x values",
    ylabel = "y values")

plot!(X,Y,
label = false)
```



There are several cases where the line plot and scatter plots are not enough for representing your data. Another type of plot that is very commonly used is bar plot. Bra plots are particularly useful for the discreet independent variable (i.e. X).

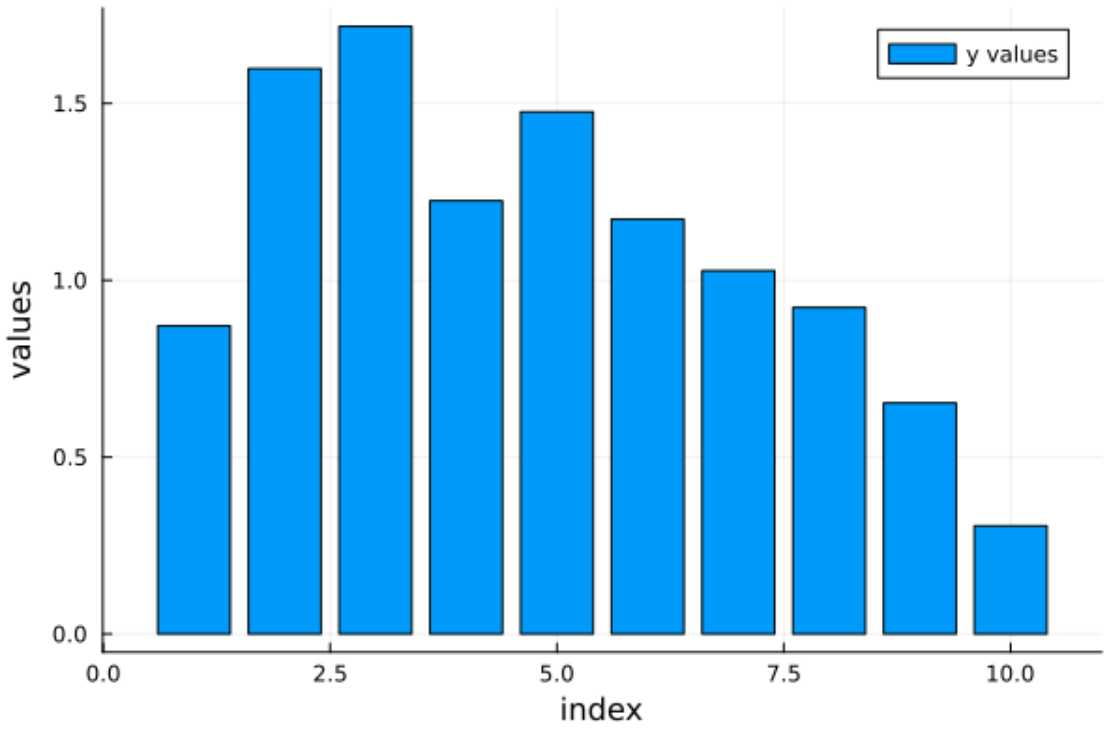
1.18 E18:

Let's display our Y data as a bar plot.

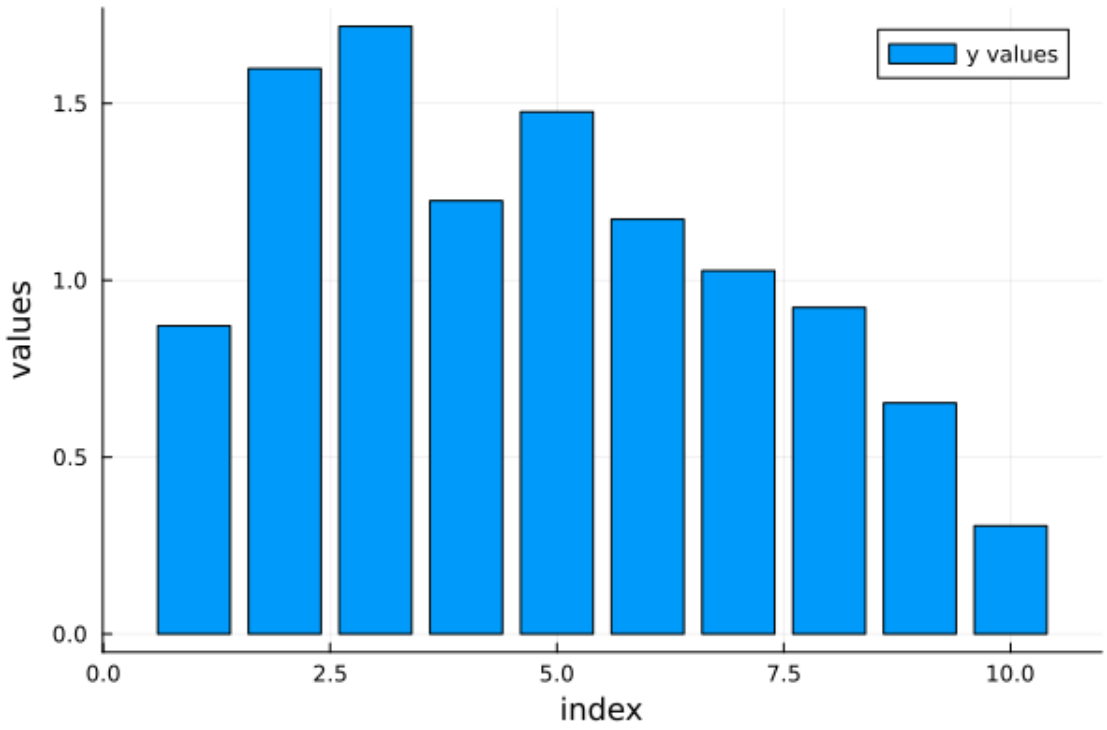
Answers to E18

For bar plots you can either use the command *bar(-)* or adjust the "st" variable in the *plot(-)*.

[ ]:
```
bar(Y,
label = "y values",
xlabel = "index",
ylabel = "values")
```



[ ]:
```
# or

plot(Y,
label = "y values",
xlabel = "index",
ylabel = "values",
st = :bar)
```



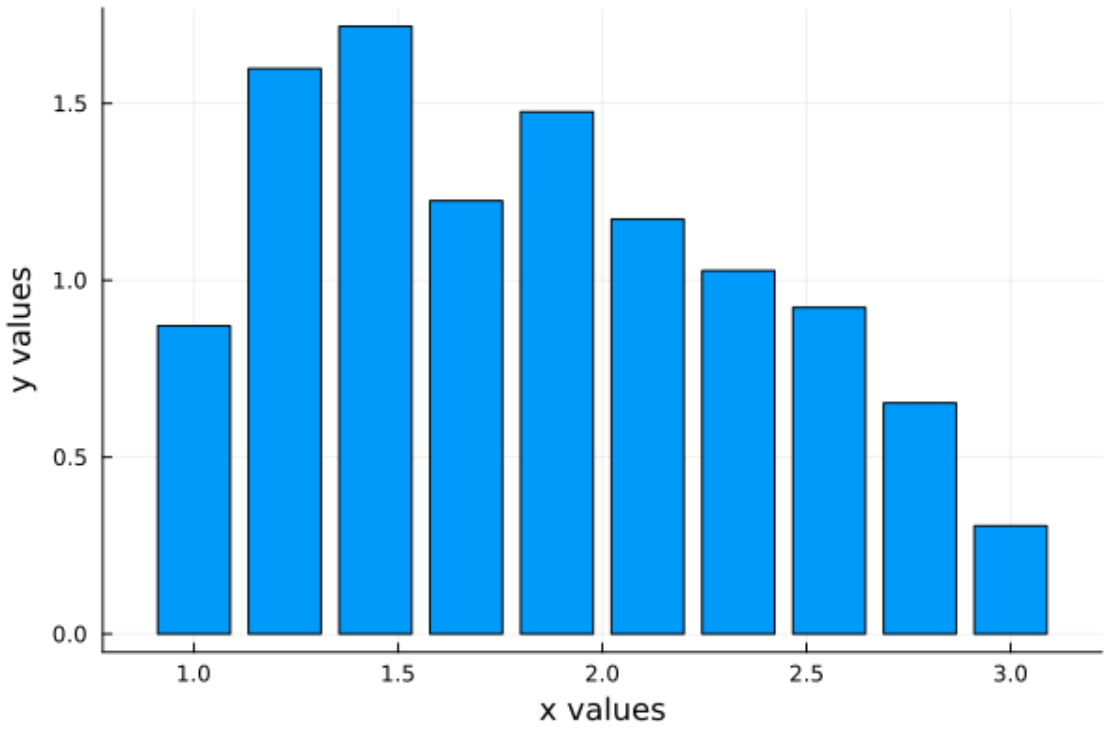
1.19 E19:

How can we adjust the x axis to represent the X values?

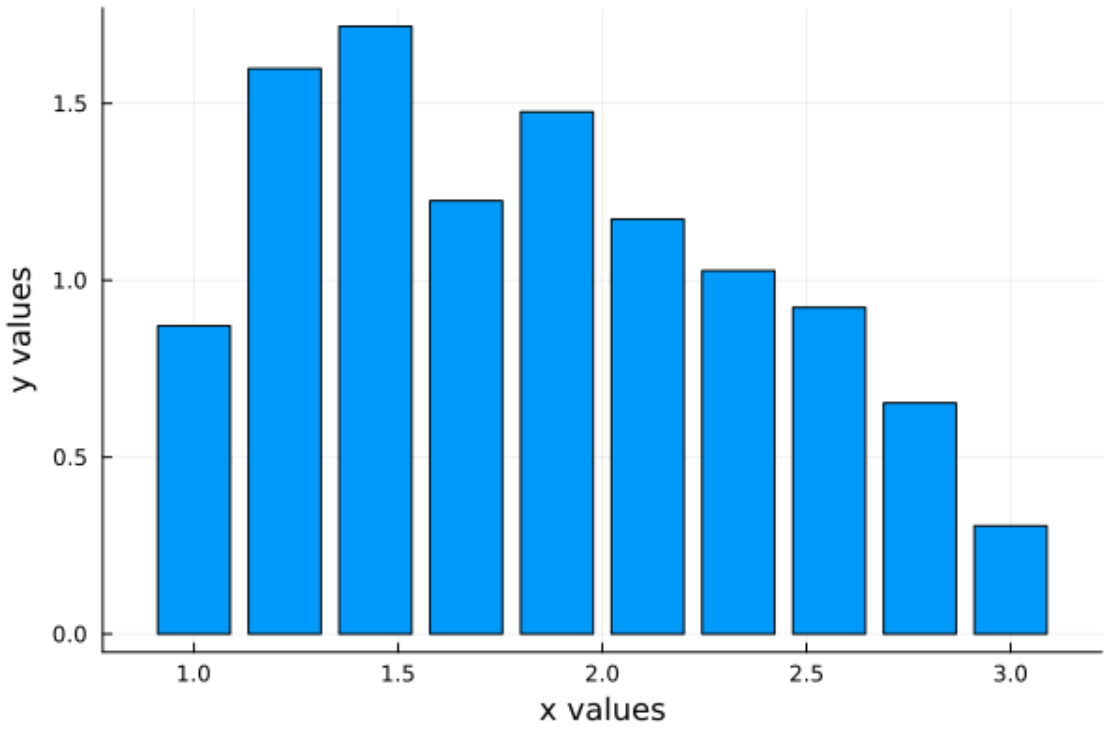
Answers to E19

For this we can provide the X to the plotting function (e.g. *bar(-)*) as a variable.

[ ]:
```
plot(X,Y,
label = false,
xlabel = "x values",
ylabel = "y values",
st = :bar)
```



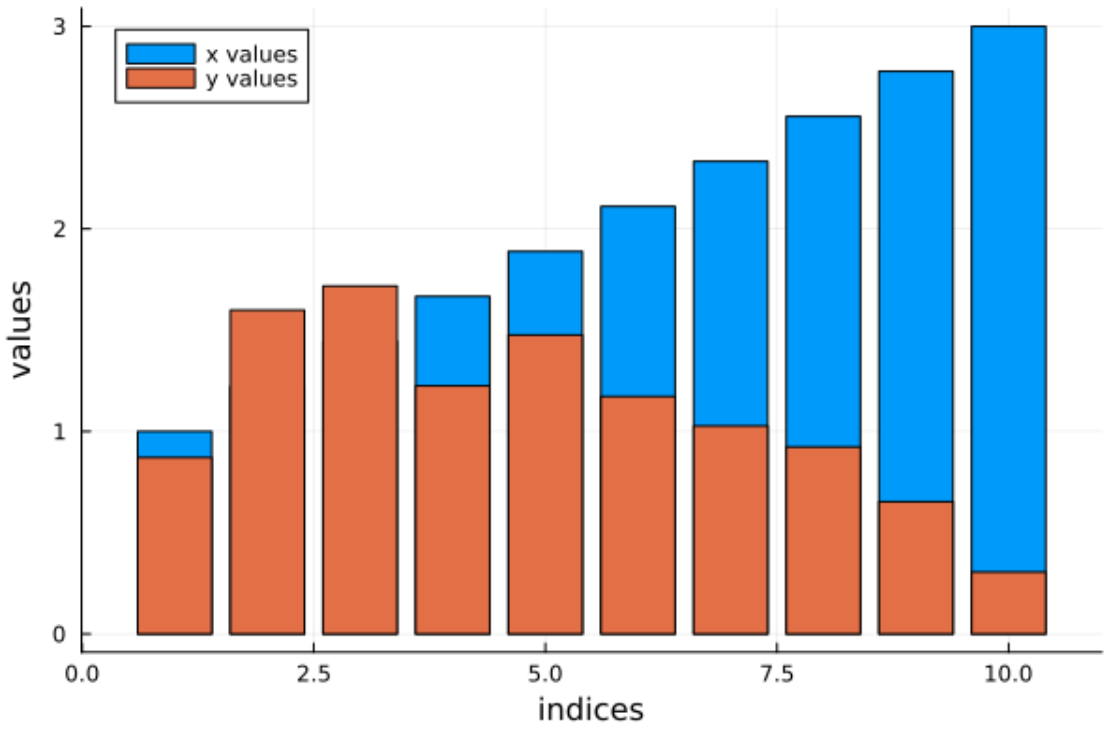
[ ]:
```
bar(X,Y,
label = false,
xlabel = "x values",
ylabel = "y values")
```



1.20 E20:

What if we want to plot multiple data series via bar plots? Let's try to plot X and Y series as two overlapping bar plots.

[ ]:
```
bar(X,
label = "x values",
xlabel = "indices",
ylabel = "values")

bar!(Y,
label = "y values")
```



The two series are overlapping and may cause some difficulties for the interpretation.

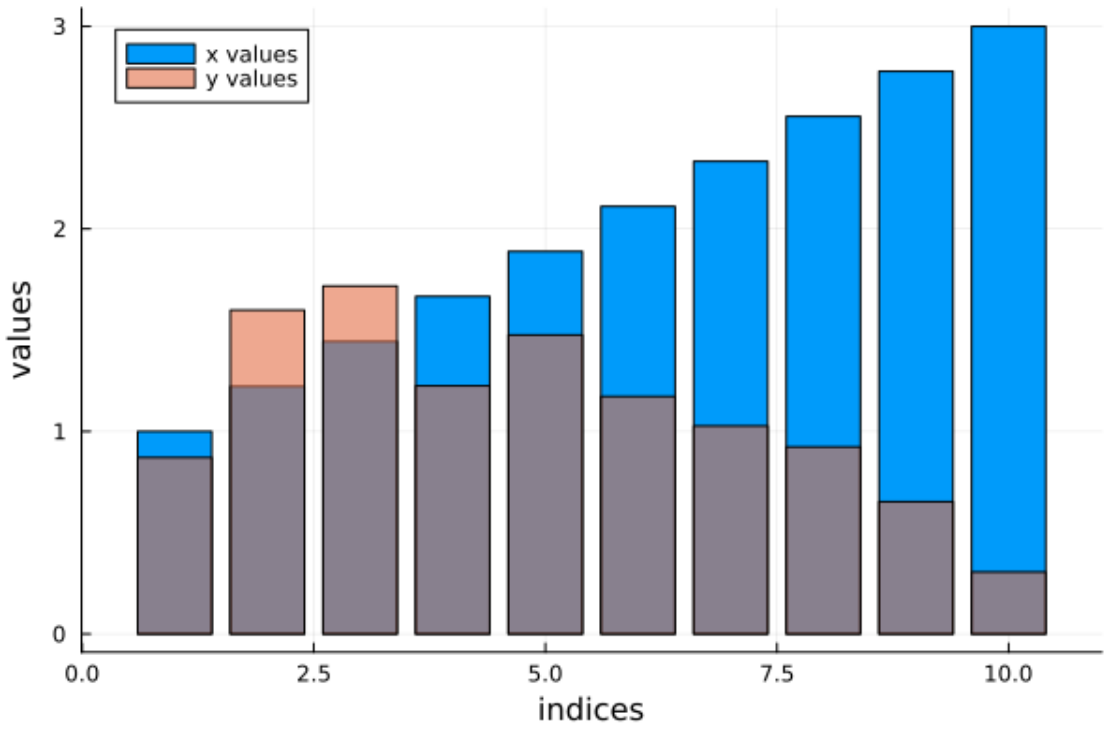
1.21 E21:

Is there a way to overcome this issue? Can we for example change the transparency of the Y?

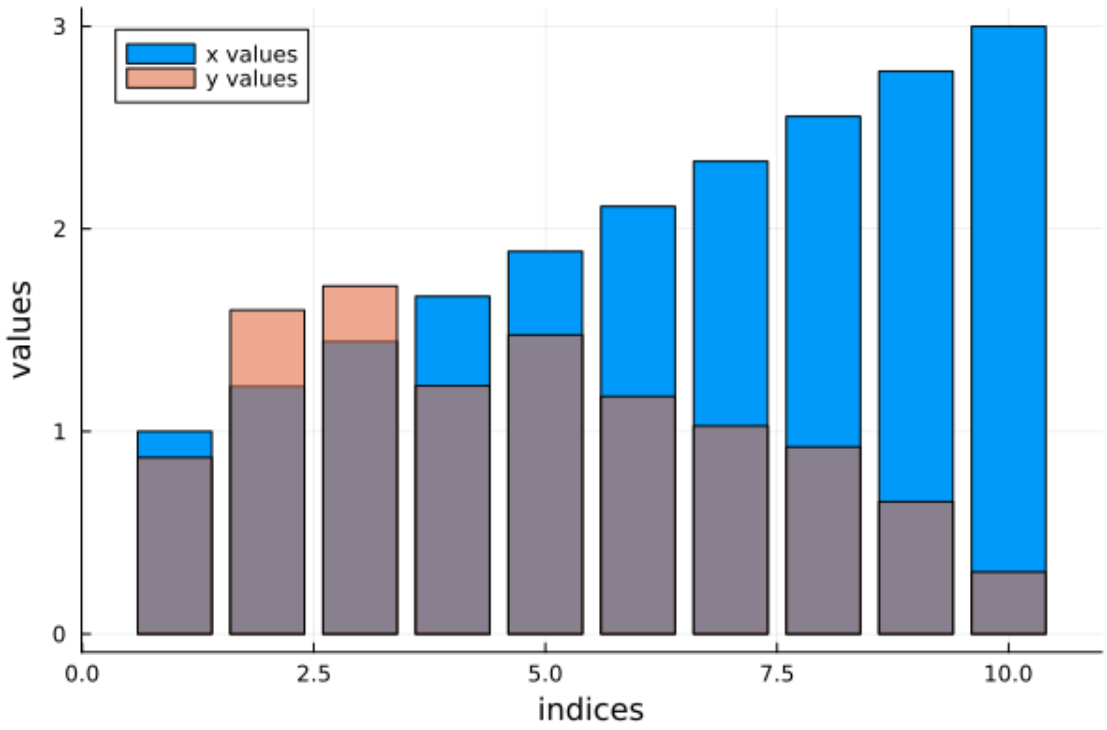
Answers to E21

You can change the transparency of a bar plot by using the parameter "fillalpha" via the functions *bar(-)* or *plot(-)*.

[ ]:
```
bar(X,
label = "x values",
xlabel = "indices",
ylabel = "values")

bar!(Y,
label = "y values",
fillalpha = 0.6)
```



[ ]:
```
plot(X,
label = "x values",
xlabel = "indices",
ylabel = "values",
st = :bar)

plot!(Y,
label = "y values",
fillalpha = 0.6,
st = :bar)
```



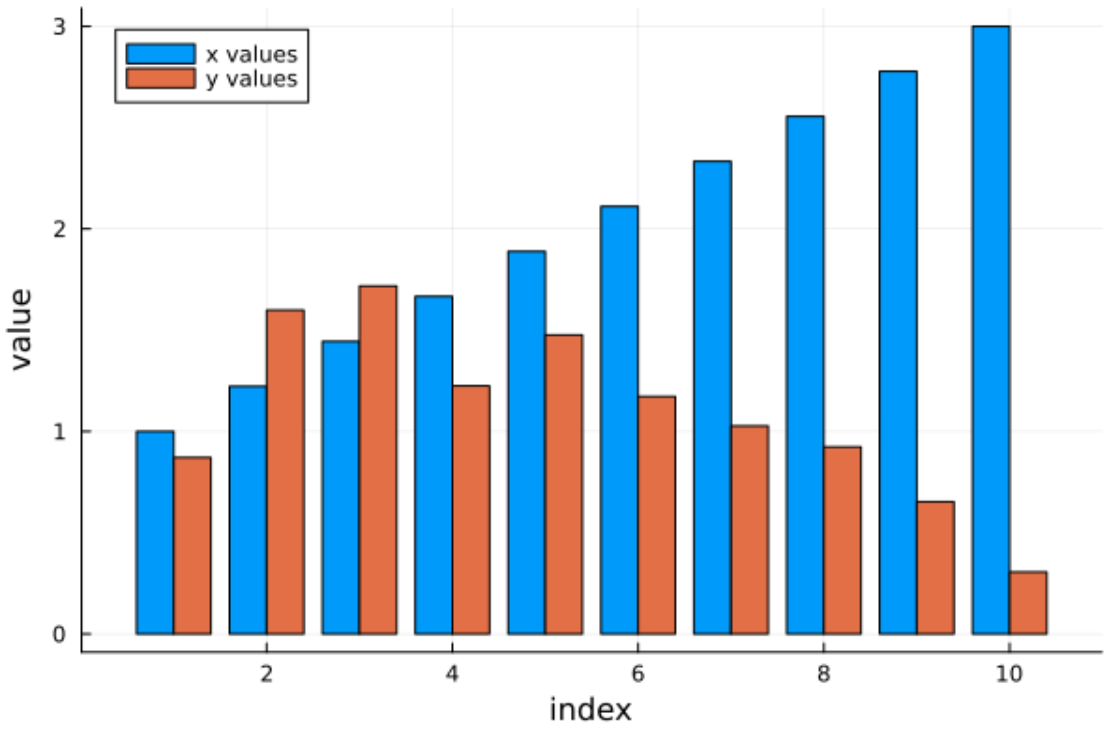
1.22 E22:

What if we want to have the bar plots sit next to each other? Let's try to plot the X related bars next to the bars for Y.

Answers to E22

For plotting the bars next to each other, you will need to use a more complete julia plot backend called StatsPlots. This backend has been exported as "sp" to avoid conflicts with the Plots.jl function. To access the functions in StatsPlots you need to do *sp.plot(-)*, for example. For this particular case you will have to use *groupedbar(-)* from StatsPlots.

[ ]:
```
sp.groupedbar([X Y],
    bar_position = :dodge,
    xlabel = "index",
    ylabel = "value",
    label = ["x values" "y values"])
```



My bar plots usually have error bars. What happened to those?

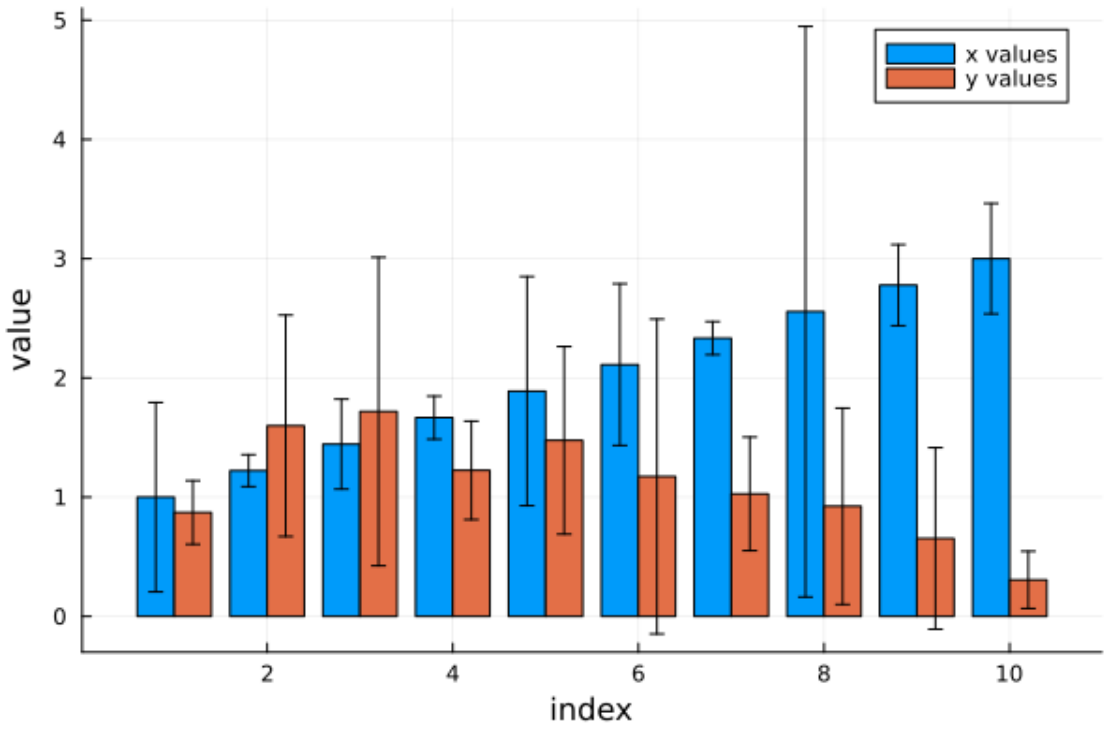
1.23 E23:

How can we add error bars to our figures?

Answers to E23

The error bars are controlled by a parameter called "yerr" for the y values or "xerr" for the x values.

[ ]:
```
y_e = randn(10,2)   # Simulating some random error values

sp.groupedbar([X Y],
    bar_position = :dodge,
    xlabel = "index",
    ylabel = "value",
    label = ["x values" "y values"],
    yerr = y_e)
```



A sub-type of bar plot that is extremely useful is the histogram. Histograms are particularly helpful for exploring the distribution of your variables.

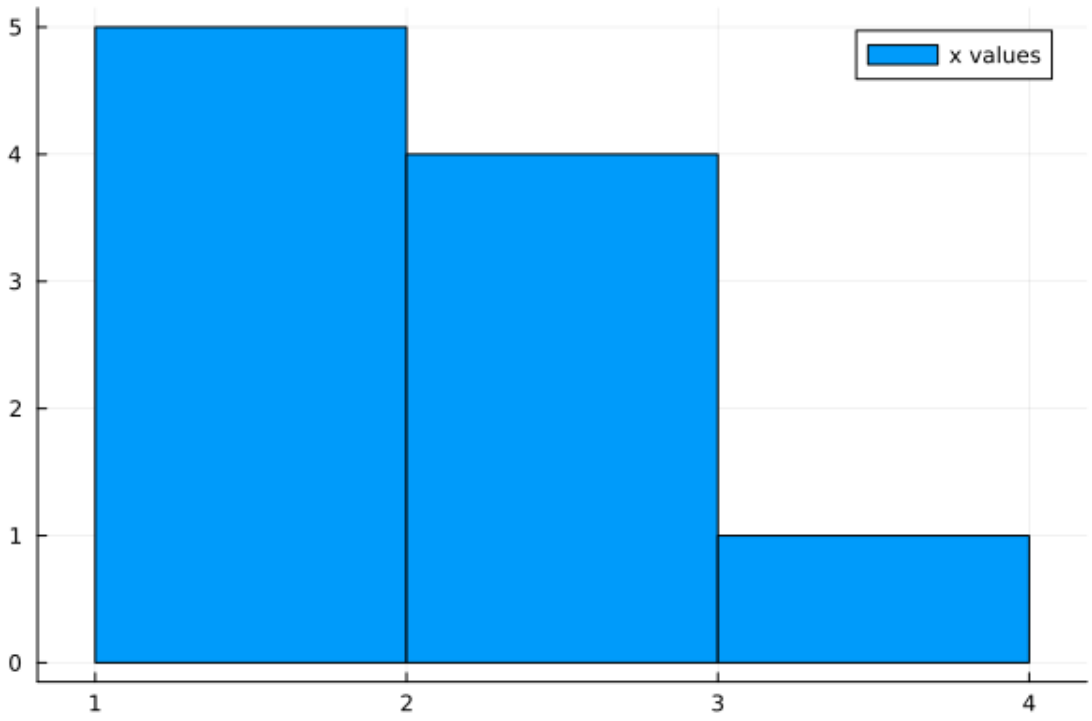
1.24 E24:

Let's plot the X values as a histogram.

Answers to E24

For plotting your data as histograms, you can use the function *histogram()* or *plot(-)* with the "st" setting.

[ ]:
```
histogram(X,
label = "x values")
```



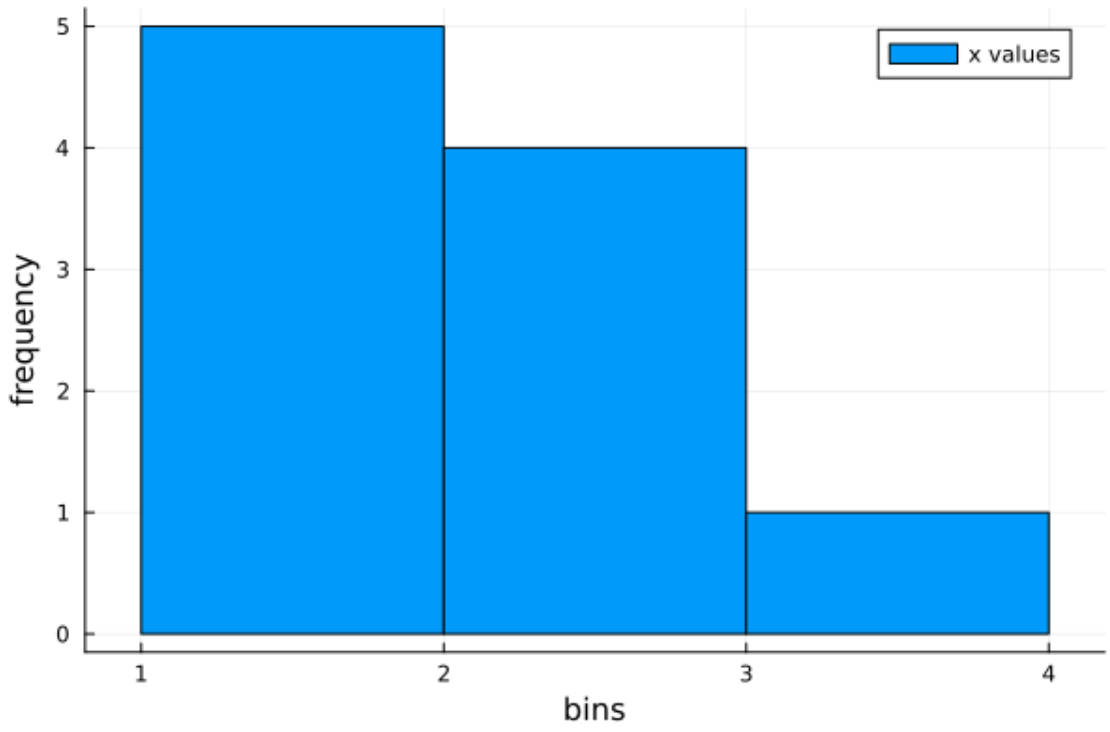
1.25 E25:

What is being shown on this plot? What are the labels of x and y axes?

Answers to E25

The x axis is the bins based on the x values. Each bin represent a range of values that are grouped together. The y axis is the count of how many data points in the X have been grouped together in each bin.

[ ]:
```
histogram(X,
label = "x values",
xlabel = "bins",
ylabel = "frequency")
```



These bins look too wide.

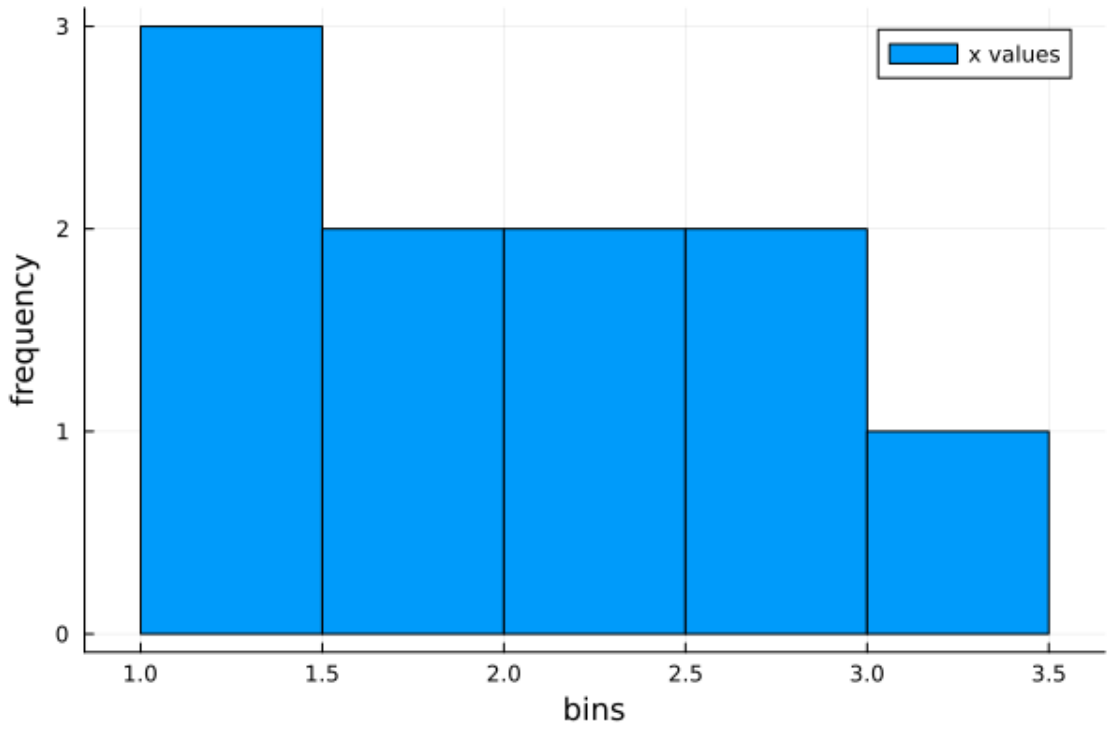
1.26 E26:

Can we change the number of the bins?

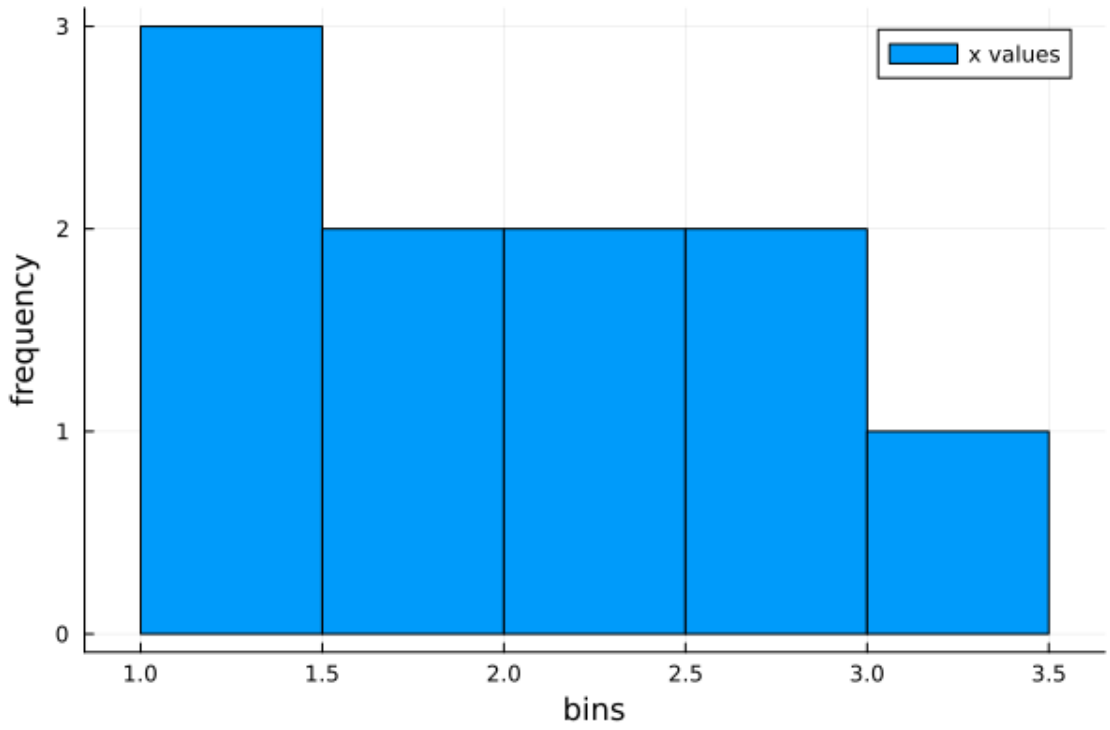
Answers to E26

The parameter to be set for the number of bins or the bin edges is "bins" that can be directly fed to the *histogram(-)* function.

[ ]:
```
histogram(X,
label = "x values",
xlabel = "bins",
ylabel = "frequency",
bins = 7)
```



[ ]:
```
b_e = range(1,3.5, length = 6)  # Setting the bin edges

histogram(X,
label = "x values",
xlabel = "bins",
ylabel = "frequency",
bins = b_e)
```



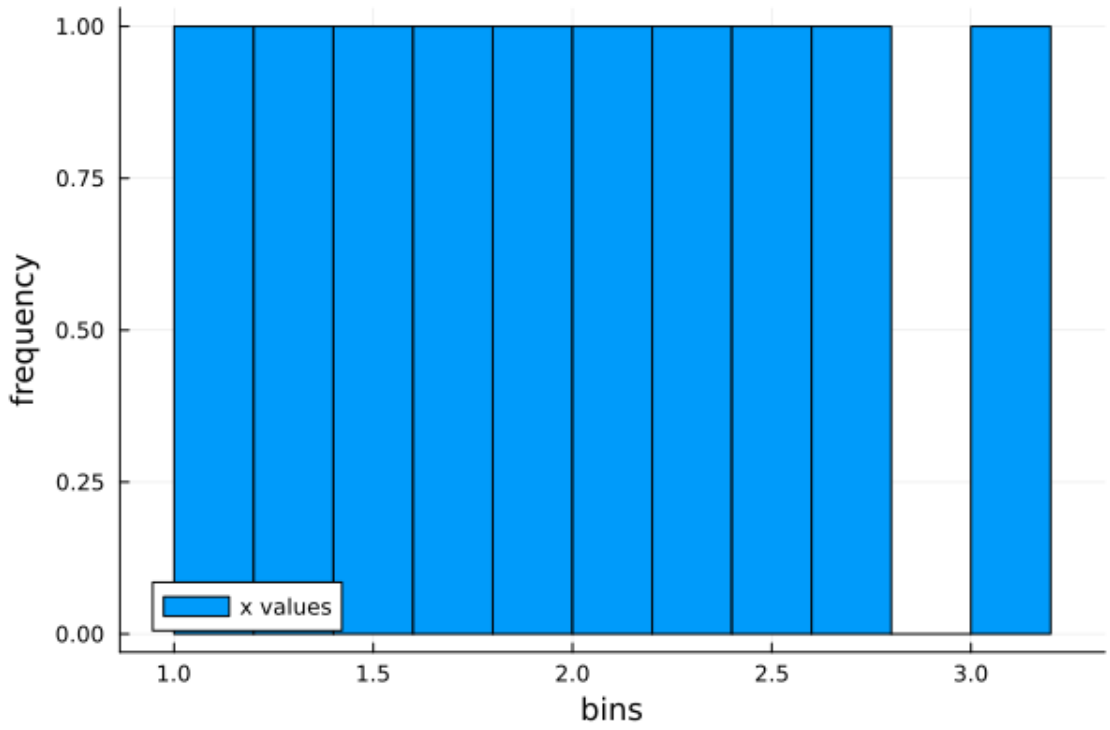
1.27 E27:

How can you choose the number of bins?

Answers to E27

It is very important to use your prior knowledge in selecting the number of bins as by choosing the wrong number you may have a completely different interpretation of the results. For example by changing the number of bins for X we have different distributions for the same dataset.

[ ]:
```
histogram(X,
label = "x values",
xlabel = "bins",
ylabel = "frequency",
bins = 10)
```



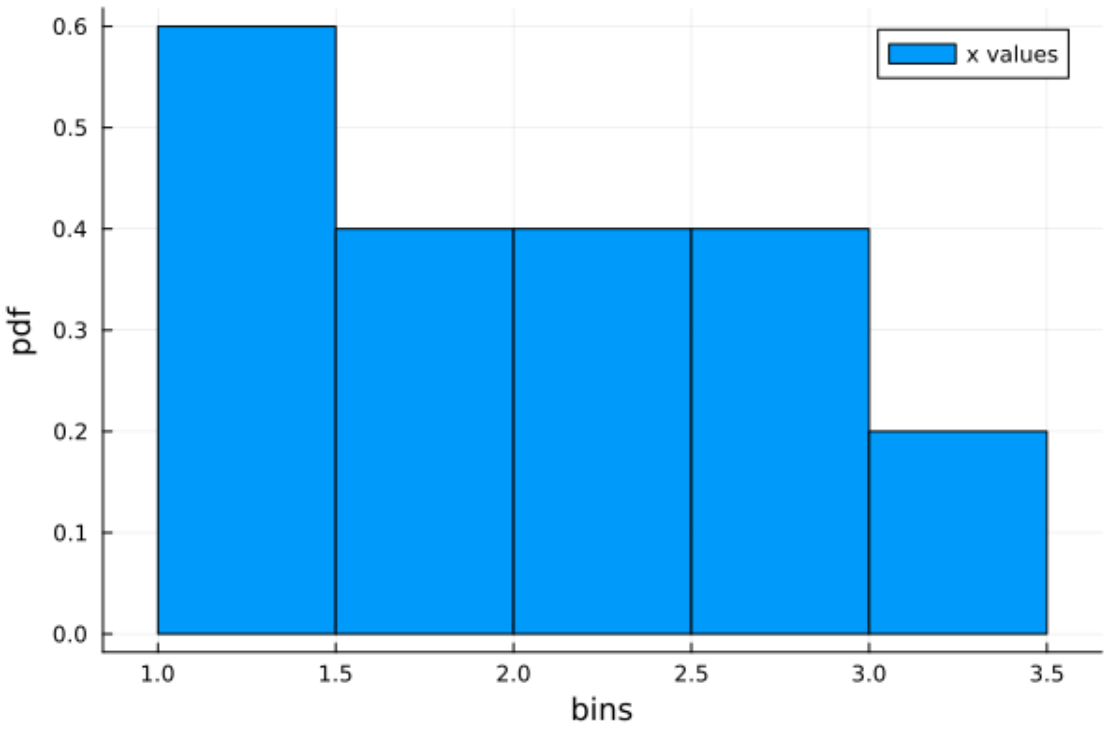
1.28 E28:

Do we have to have our y axis as a frequency? Can we have it as a probability?

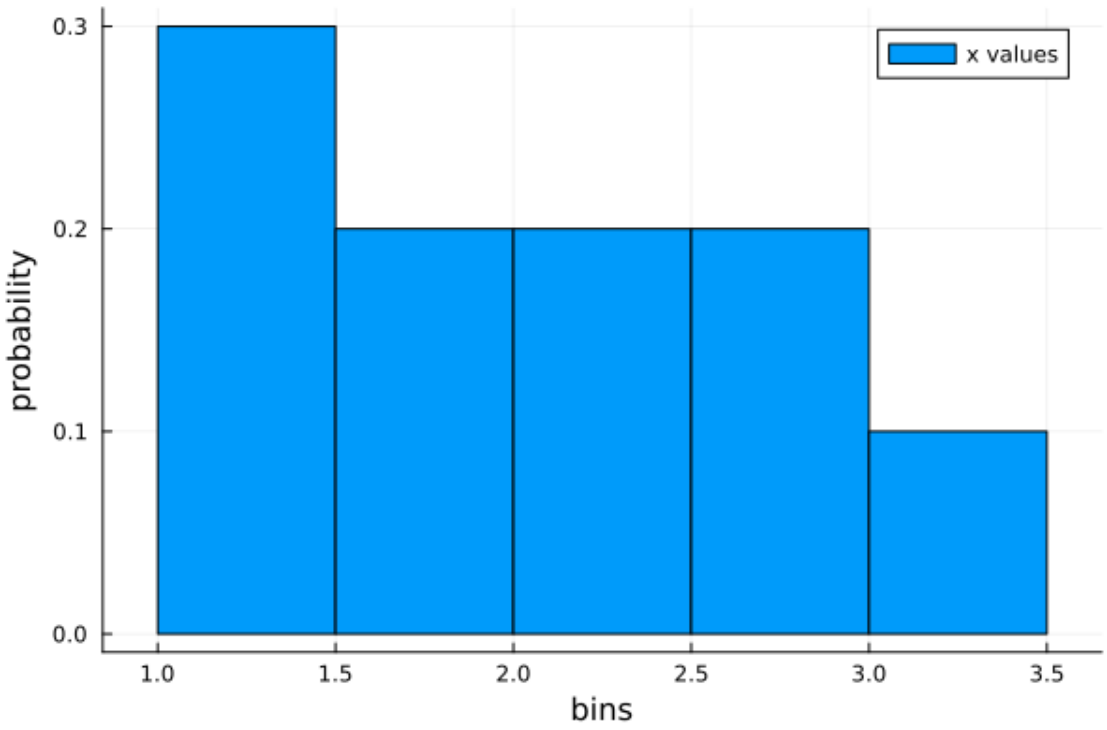
Answers to E28

You can perform different types of data normalization for your histograms using the parameter "norm". The two most commonly used options for this parameter are :pdf (i.e. sum of all areas set to one) and :probability (i.e. sum of all intensities are set to one).

[ ]:
```
histogram(X,
label = "x values",
xlabel = "bins",
ylabel = "pdf",
bins = 7,
norm = :pdf)
```



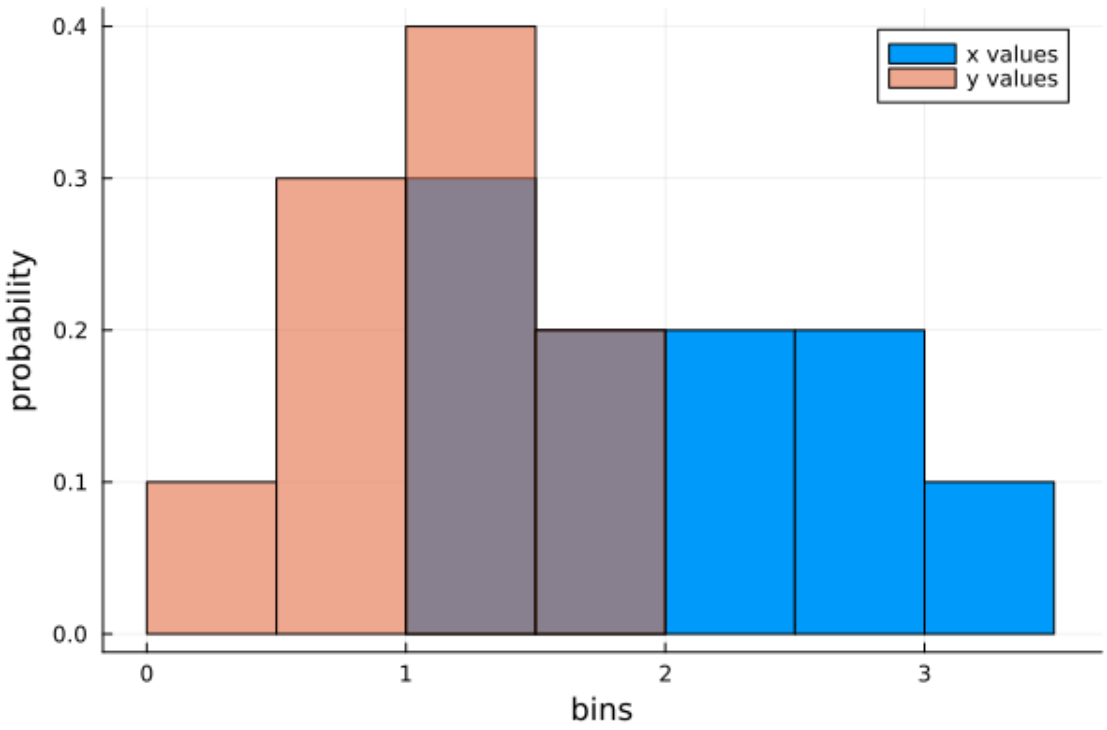
[ ]:
```
histogram(X,
label = "x values",
xlabel = "bins",
ylabel = "probability",
bins = 7,
norm = :probability)
```



1.29 E29:

What about multiple histograms? Can we overly two or more histograms on top of each other?

[ ]:
```
histogram(X,
label = "x values",
xlabel = "bins",
ylabel = "probability",
bins = 7,
norm = :probability)

histogram!(Y,
label = "y values",
bins = 5,
norm = :probability,
fillalpha = 0.6)
```



When overlaying histograms you need to use the "fillalpha" parameter to set the transparency of each distribution.

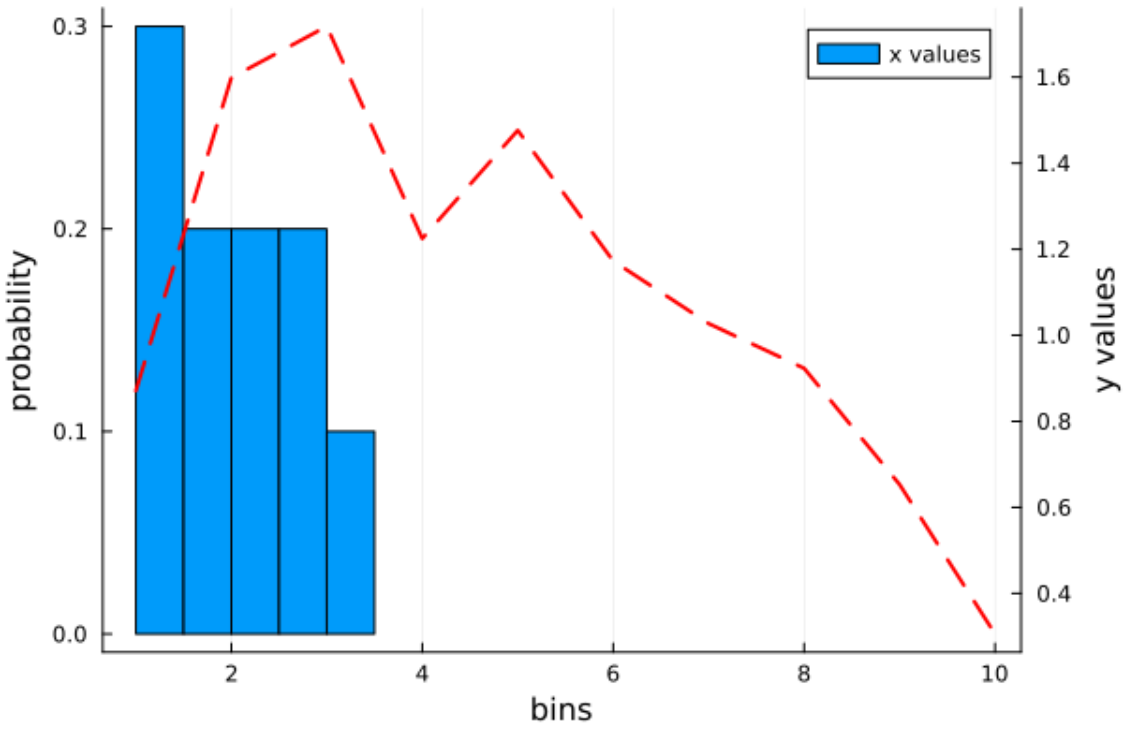
1.30 E30:

Can we combine the histograms with line plots for example?

Answers to E30

This is only possible if the second series has also probability or frequency unit or you will use the second y axis.

[ ]:
```
histogram(X,
label = "x values",
xlabel = "bins",
ylabel = "probability",
bins = 7,
norm = :probability)

plot!(DataSci4Chem.twinx(),Y,
    ylabel = "y values",
    label = false,
    lc = :red,
    lw = 2,
    ls = :dash)
```



As you can see, this figure is not very informative due to the inadequacy of the dataset used. However, the workflow used can be employed for your future figures.

We working with multi dimension data, heatmaps are very helpful as they can facilitate visualization of 3D data no problem.

1.31 E31:

Let's generate a dataset adequate for a heatmap. Here we need a matrix rather than a vector (e.g. *Data_{10 x 20}*).

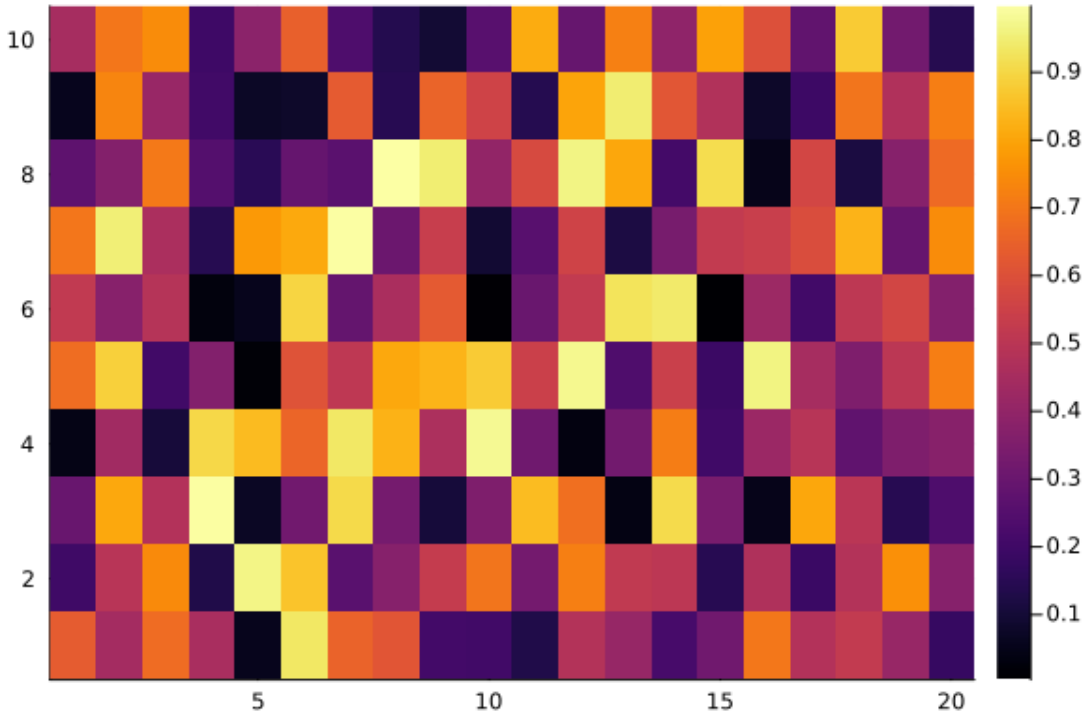
[ ]:
```
Data = rand(10,20)
```

```
10×20 Matrix{Float64}:
 0.631791   0.441749  0.673779  0.454039   …  0.519289  0.410241  0.175355
 0.196891   0.492475  0.743846  0.126778      0.48016   0.755603  0.368442
 0.293776   0.806727  0.479389  0.995072      0.499392  0.14207   0.232464
 0.0406632  0.435228  0.102667  0.902684      0.275265  0.349772  0.371569
 0.676588   0.88552   0.200903  0.359258      0.348562  0.502082  0.716721
 0.516264   0.371004  0.485218  0.0258432  …  0.506573  0.563886  0.360314
 0.697124   0.951652  0.460025  0.141231      0.828646  0.288793  0.748257
 0.269403   0.361994  0.705705  0.246731      0.11371   0.367886  0.668975
 0.053437   0.731331  0.411652  0.200793      0.695249  0.471707  0.716803
 0.446286   0.698467  0.748054  0.19338       0.876131  0.320015  0.139307
```

1.32 E32:

How can we generate a heatmap based on the *Data*?

```
[ ]: heatmap(Data,
         label = "random data")
```



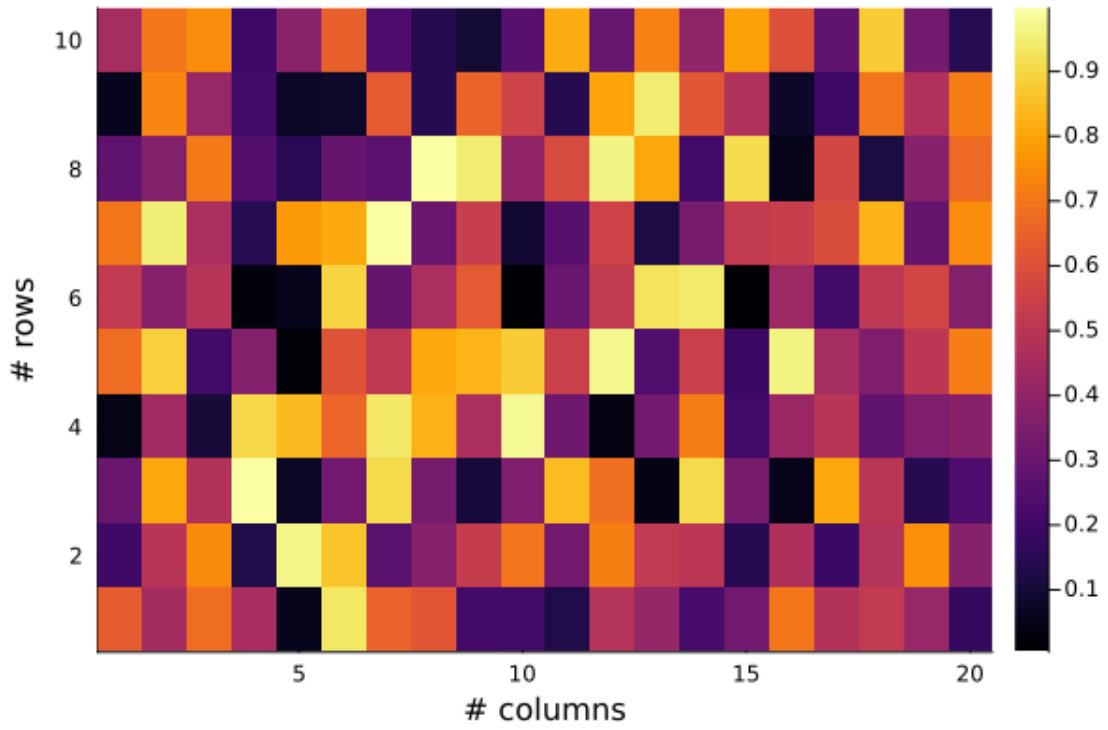
1.33 E33:

What are the labels of x and y axes? Add the labels to this plot.

Answers to E33

Both axes are the indices of *Data*. For example the x axis is the column number and the y axis is the row number.

```
[ ]: heatmap(Data,
         label = "random data",
         xlabel = "# columns",
         ylabel = "# rows")
```



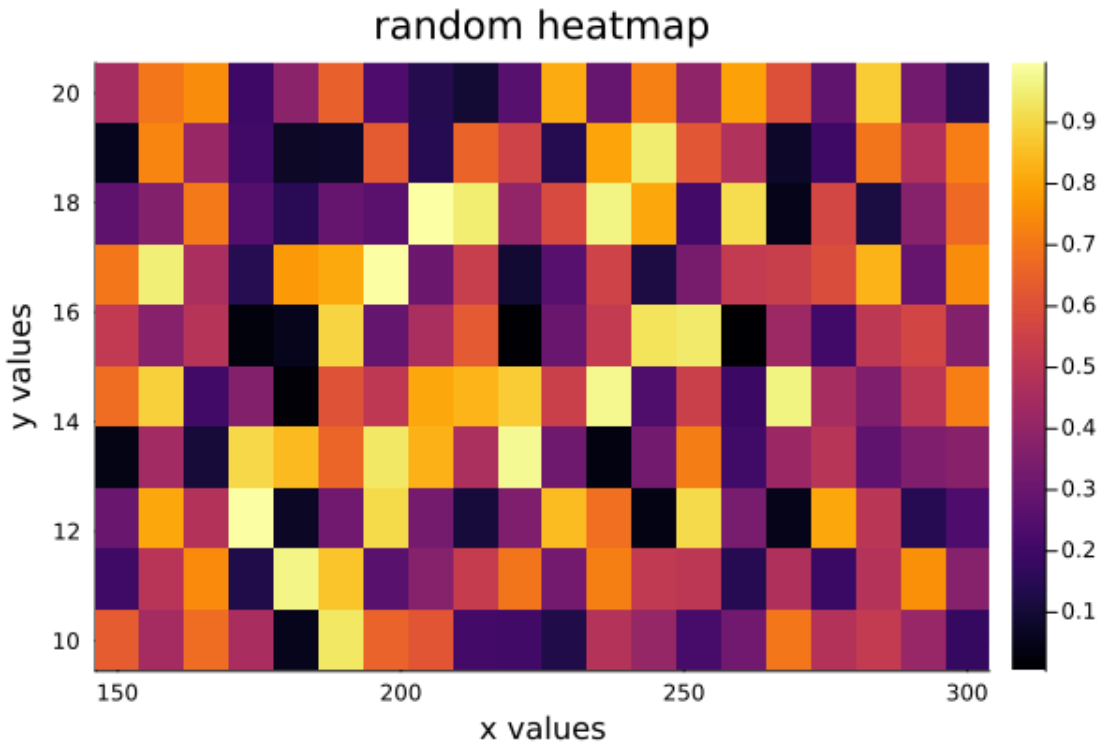
1.34 E34:

If we were to actually add the true x and y values to this plot, what would be the sizes of X and Y? Generate the appropriate X and Y between 10:20 and 150:300, respectively.

Answers to E34

The Y must be a vector of 10 entries between 10 and 20 while the X needs to be a vector of 20 entries between 150 and 300.

[ ]:
```
Y1 = range(10,20,length = size(Data,1))
X1 = range(150,300,length = size(Data,2))

heatmap(X1,Y1,Data,
    label = "random data",
    xlabel = "x values",
    ylabel = "y values",
    title = "random heatmap")
```



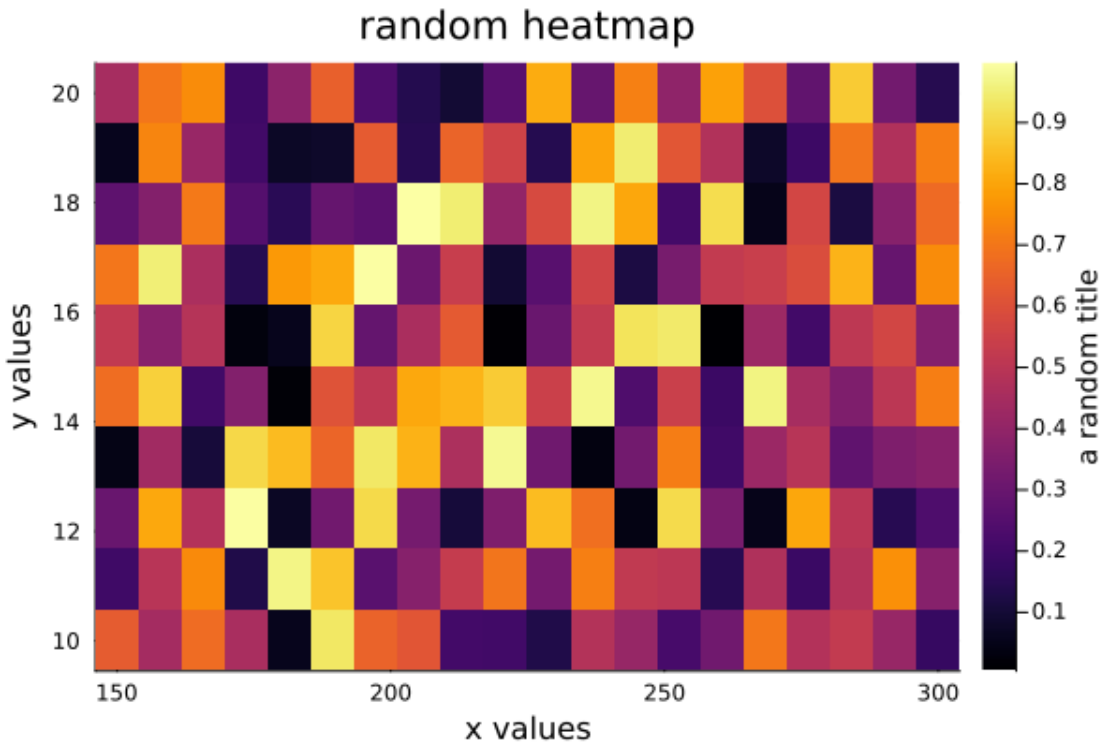
1.35 E35:

I do not understand the what these colors are? Can you label the colorbar?

Answers to E35

You can add a colorbar title to your plot using the attribute "colorbar_title". This attribute takes a string as the variable.

[ ]:
```
heatmap(X1,Y1,Data,
    label = "random data",
    xlabel = "x values",
    ylabel = "y values",
    title = "random heatmap",
    colorbar_title = "a random title")
```



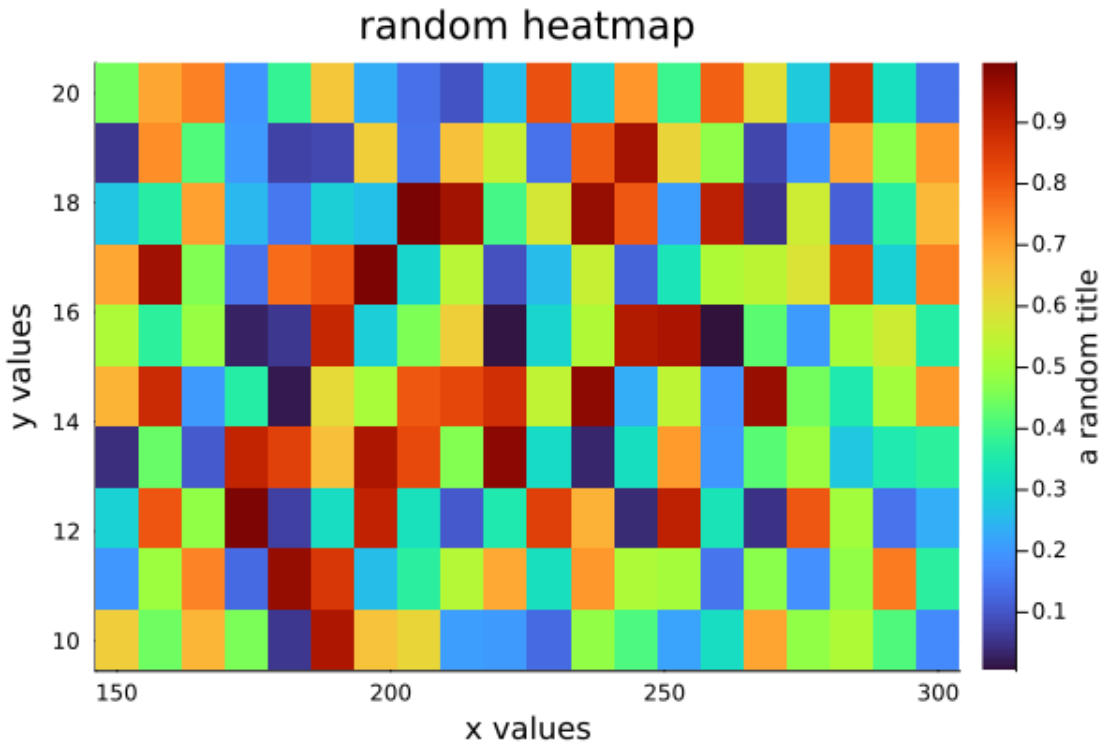
1.36 E36:

I am also color blind and cannot see your colors well. Can you change the color scheme of your plot?

Answers to E36

The color scheme is usually set by an attribute called "cmap". There are several color schemes implemented within *Plots.jl* package.

[ ]:
```
heatmap(X1,Y1,Data,
    label = "random data",
    xlabel = "x values",
    ylabel = "y values",
    title = "random heatmap",
    colorbar_title = "a random title",
    cmap = :turbo)
```



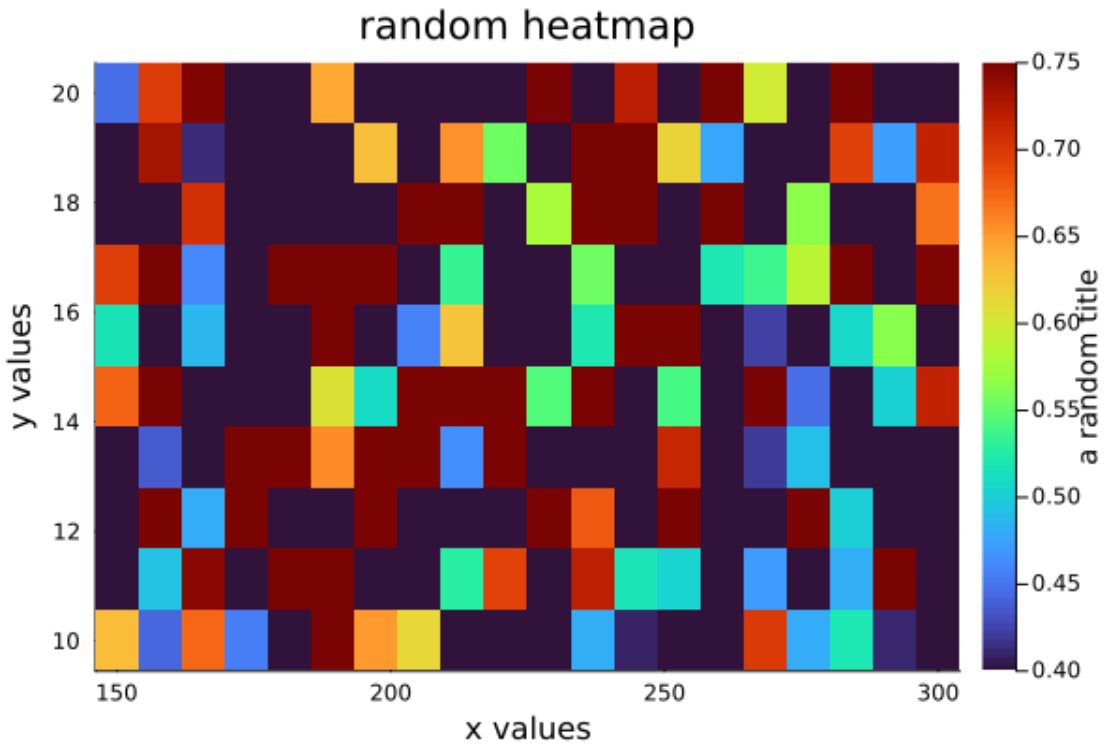
1.37 E37:

Much better! But maybe you can limit the color range between 0.4 and 0.75.

Answers to E37

To set the limits of color range, you can use the attribute "clim". This parameter takes a tuple (e.g. (0.4,0.75)) of the limits as input.

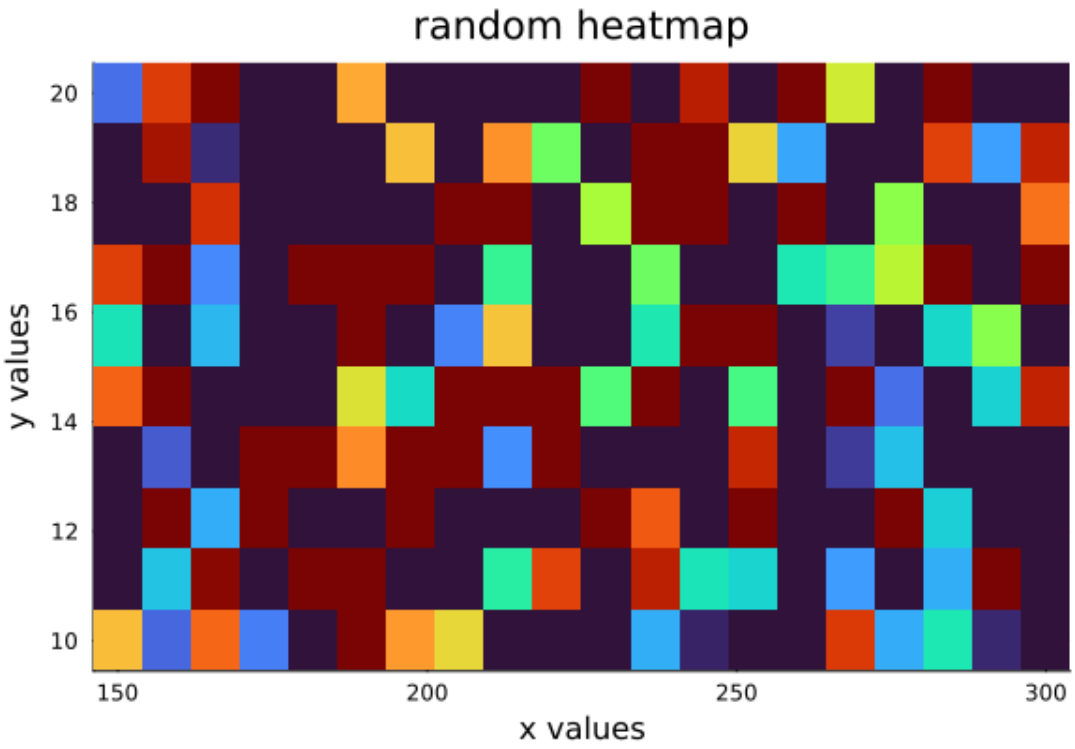
[ ]:
```
heatmap(X1,Y1,Data,
    label = "random data",
    xlabel = "x values",
    ylabel = "y values",
    title = "random heatmap",
    colorbar_title = "a random title",
    cmap = :turbo,
    clim = (0.4,0.75))
```



1.38 E38:

I think the colorbar is very distracting. Let's remove it completely.

[ ]:
```
heatmap(X1,Y1,Data,
    label = "random data",
    xlabel = "x values",
    ylabel = "y values",
    title = "random heatmap",
    cbar = :none,
    colorbar_title = "a random title",
    cmap = :turbo,
    clim = (0.4,0.75))
```



1.39 E39:

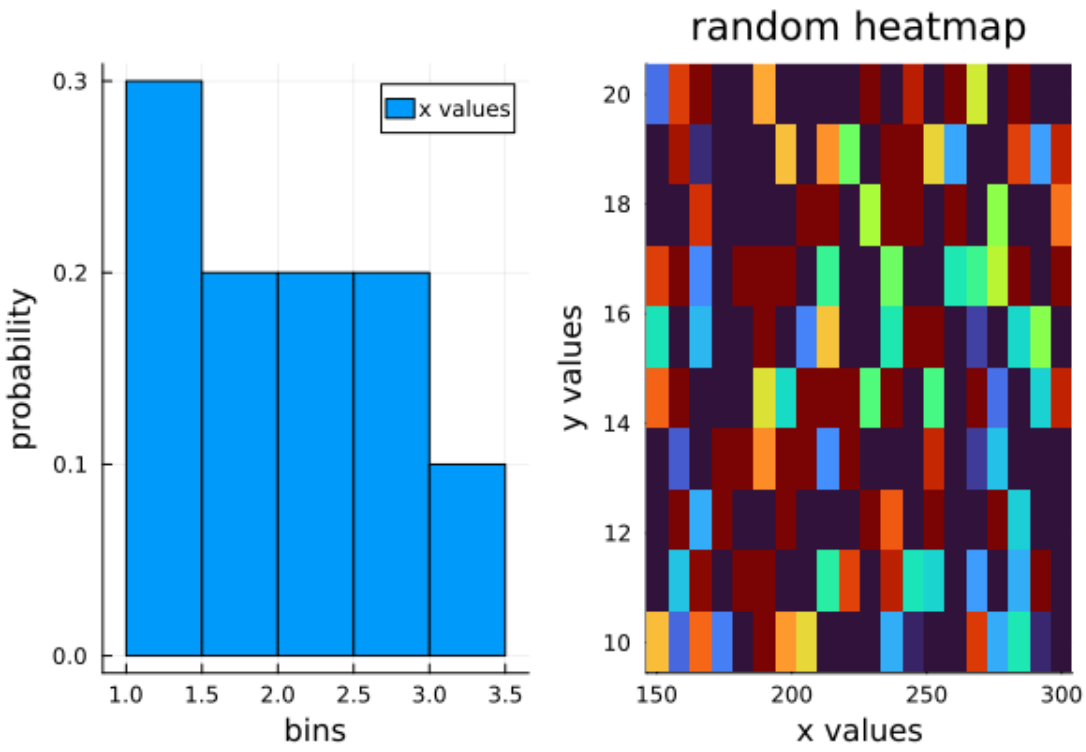
What if I want to see the bar plot and the heatmap one next to another. Is there a way to combine multiple figures into one frame?

Answers to E39

To have multiple figures in a single frame you can use subplots. For subplots, you need to assign each plot to a variable. Then you can use the combination of *plot(-)* and the attribute "layout" to generate subplots.

[ ]:
```
p1 = histogram(X,
    label = "x values",
    xlabel = "bins",
    ylabel = "probability",
    bins = 7,
    norm = :probability)

p2 = heatmap(X1,Y1,Data,
    label = "random data",
    xlabel = "x values",
    ylabel = "y values",
    title = "random heatmap",
    cbar = :none,
```

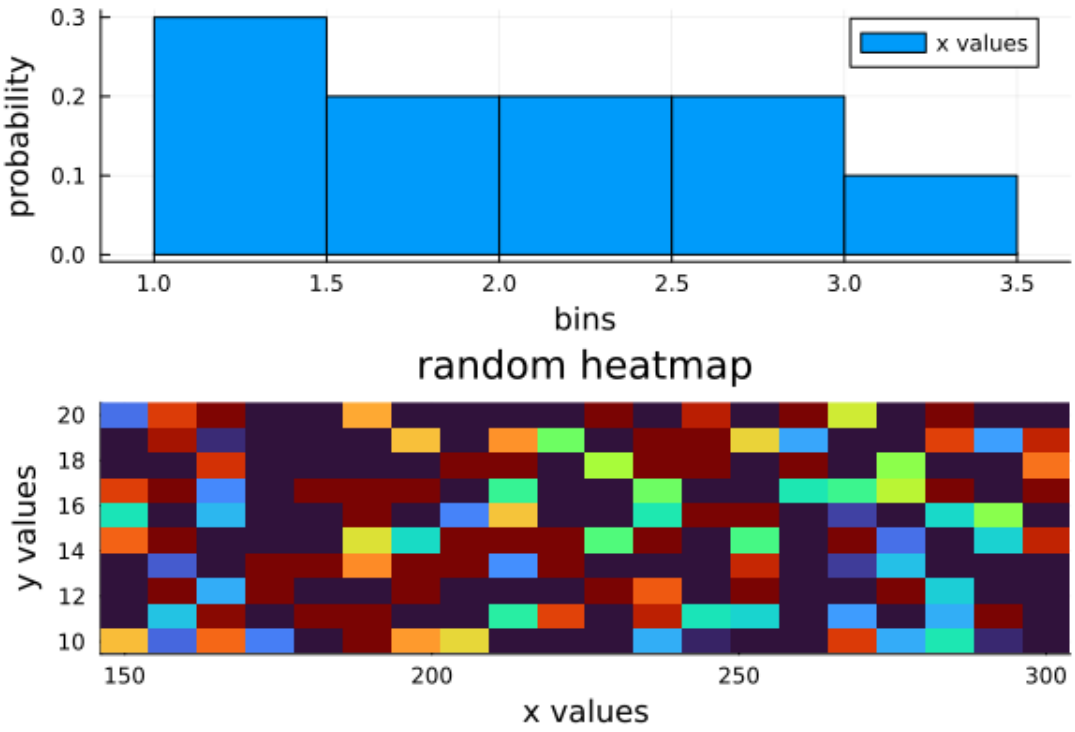
```
    colorbar_title = "a random title",
    cmap = :turbo,
    clim = (0.4,0.75))

plot(p1,p2)
```



1.40 E40:

Well this is nice but what if we want to change the layout of these plots. Let's try to make a 2x1 plot grid.

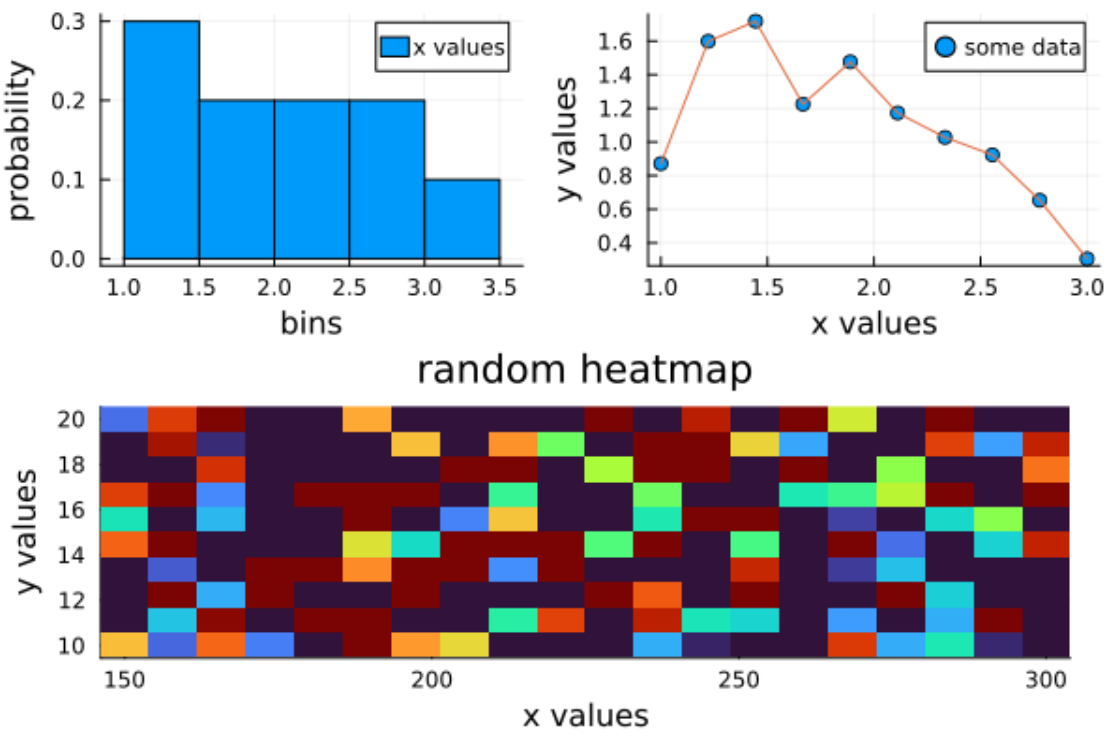
[ ]:
```
plot(p1,p2,layout = (2,1))
```



1.41 E41:

Great! But I also want to see the scatter plots of X and Y in the top panel of this figure. Let's generate a figure with three total panels two on top and one in the bottom.

[ ]:
```
p3 = scatter(X,Y,
    label = "some data",
    xlabel = "x values",
    ylabel = "y values")

plot!(X,Y,
label = false)

plot(plot(p1,p3),p2,
    layout = (2,1))
```



1.42 E42:

This is a nice plot! Now we need to save it either as a pdf or png. How can we do that?

[ ]:
```
savefig("test_fig.png")
```

```
"/Users/saersamanipour/Desktop/dev/pkgs/DataSci4Chem.jl/Notebooks/test_fig.png"
```

[ ]:
```
# or
savefig("test_fig.pdf")
```

```
"/Users/saersamanipour/Desktop/dev/pkgs/DataSci4Chem.jl/Notebooks/test_fig.pdf"
```

2 Exercises

Below you have a set of exercises that you can use for practicing your plotting skills.

2.1 EX1:

Plot multiple series with different numbers of points. Mix arguments that apply to all series (marker/markersize) with arguments unique to each series (colors). Special arguments line, marker, and fill will automatically figure out what arguments to set.

2.2 EX2:

Regenerate this plot using random data. (Hint: the X and Y are highly correlated)