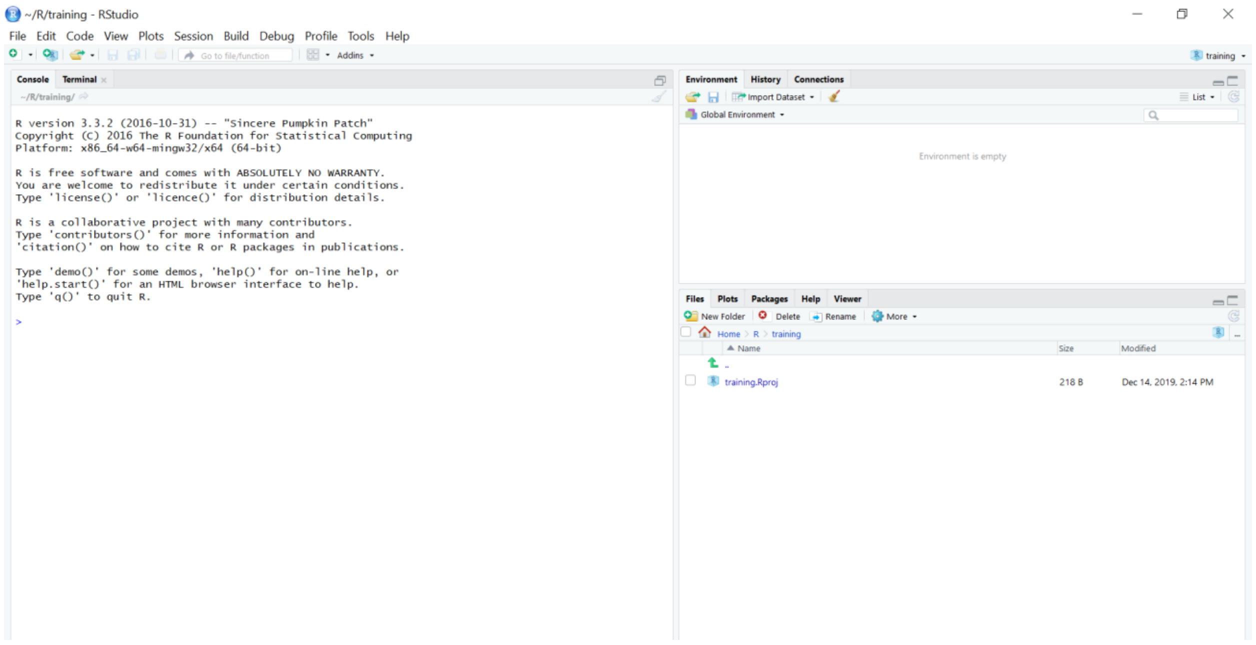
Mini-Tutorials for Creating Graphs in R

Bar Graphs

The examples below are intended to instruct you to create statistical graphs in R with minimal initial training in R. You should be able to follow the example codes to obtain graphs by modifying the included code. Some examples (and a key) will be included at the end of the document for practice. The screenshots I will show of the environment use R Studio, which is a free program you can find online. Other R environments will look different, but probably have similar functionality.

When you open up a new project environment in R Studio, it looks like this.



```
R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 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.

>
```

The command line environment is on the left. Images when we construct them will appear on the bottom right. As we add variables, they will appear in the list at the top right (name, dimensions and samples will display, which is useful for checking that you didn't skip entries when entering data by hand).

We are going to start by creating a bar graph (the same as a column graph in Excel). The data should be pre-summarized for this example. This method will work for pre-summarized histogram data, too.

Copy the commands shown into the command line.

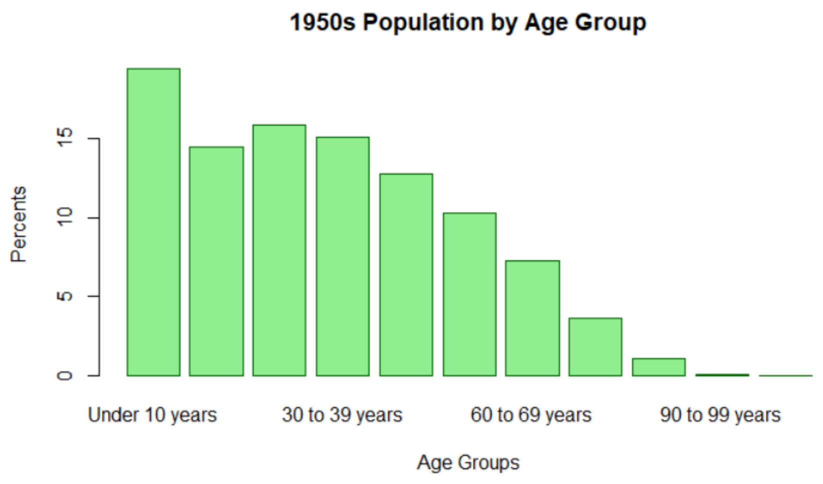
Step 1. Enter the data to be plotted in vector form. You'll need one list for the frequencies, and one for the bins or categories. If you use words for the categories, don't forget the quotation marks. The frequency data can be counts or expressed as percentages, whichever you prefer.

```
percents<-c(19.39,14.43,15.88,15.09,12.77,10.26,7.28,3.64,1.06,0.07,0)
ages<-c("Under 10 years","10 to 19 years","20 to 29 years","30 to 39 years","40 to 4
9 years","50 to 59 years","60 to 69 years","70 to 79 years","80 to 89 years","90 to 99
years", "100 years and older")
```

Be sure to check, if you enter the data in by hand, that the lengths of your vectors are the same.

Step 2. Graph the bar plot.

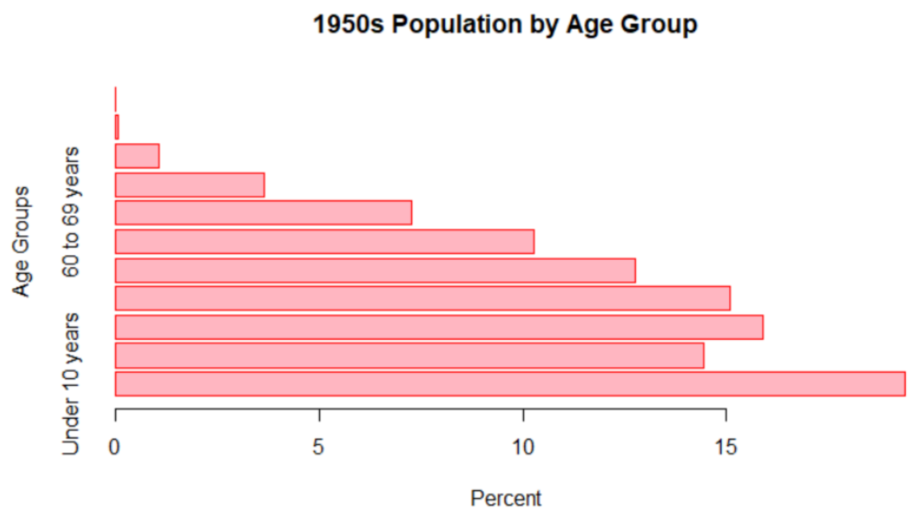
```
barplot(percents,names.arg=ages,xlab="Age Groups",ylab="Percents",col="lightgreen",
main="1950s Population by Age Group",border="darkgreen")
```



Adjust the colors to taste. Notice that since the names were too long to all fit, the graph spaced them out and only showed some of the labels so that they remained readable. This is okay if the data is ordered (as in a histogram). It's more problematic when there is no sensible order and skipping over values misses important information.

If you want the bars to be horizontal, you can do that too.

```
barplot(percents,names.arg=ages,xlab="Percent",ylab="Age Groups",col="lightpink",
main="1950s Population by Age Group",border="red", horiz=TRUE)
```



It’s possible to turn the labels sidewise with the parameter “las=2”, to be perpendicular to the axis, but if the labels are very long, they will run off the graph without further adjustments.

It’s possible to create stacked or cluster bar graphs as well, though this is a little more complicated.

How is the data organized?

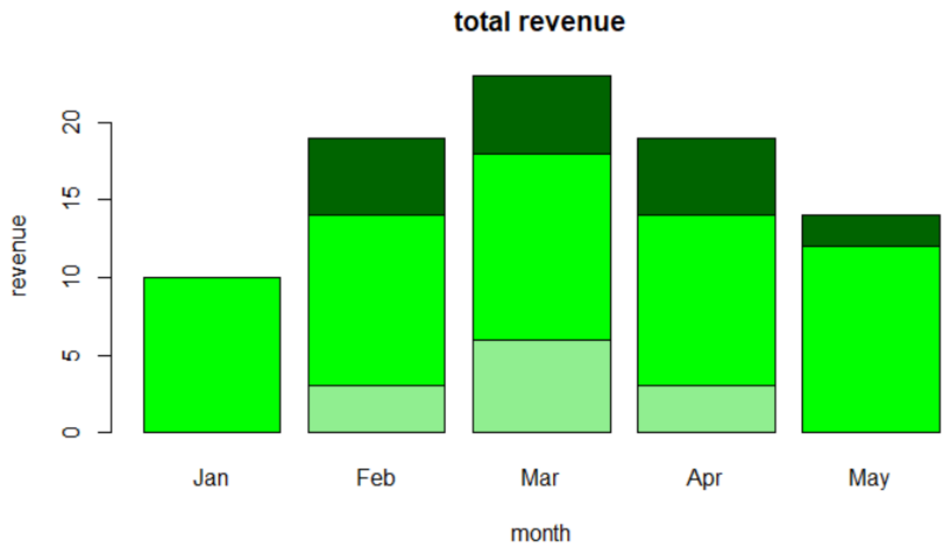
```
colors = c("lightgreen","green","darkgreen")
months <- c("Jan","Feb","Mar","Apr","May")
source <- c("NOVA","CFA","ECC")
```

Then enter the data values for the chart. You can create separate lists and then add them into the matrix, but this works.

```
Values <- matrix(c(0,3,6,3,0,10,11,12,11,12,0,5,5,5,2), nrow = 3, ncol = 5, byrow = TRUE)
```

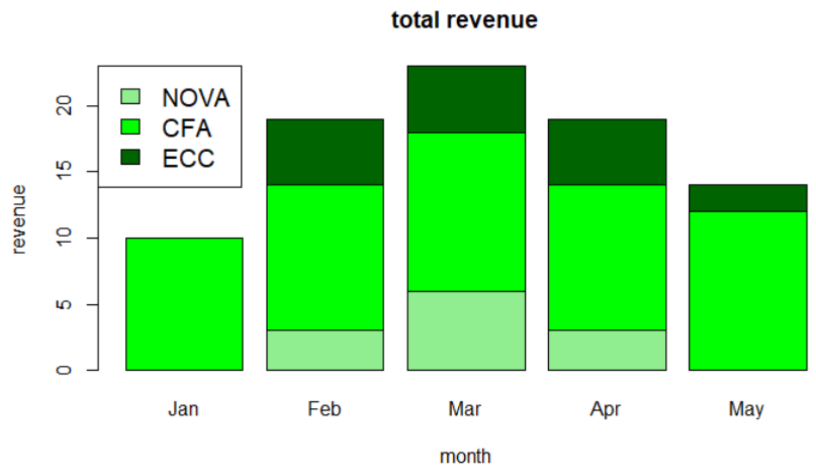
And then plot.

```
barplot(Values, main = "total revenue", names.arg = months, xlab = "month", ylab = "rev
enue", col = colors)
```



You can also add a legend.

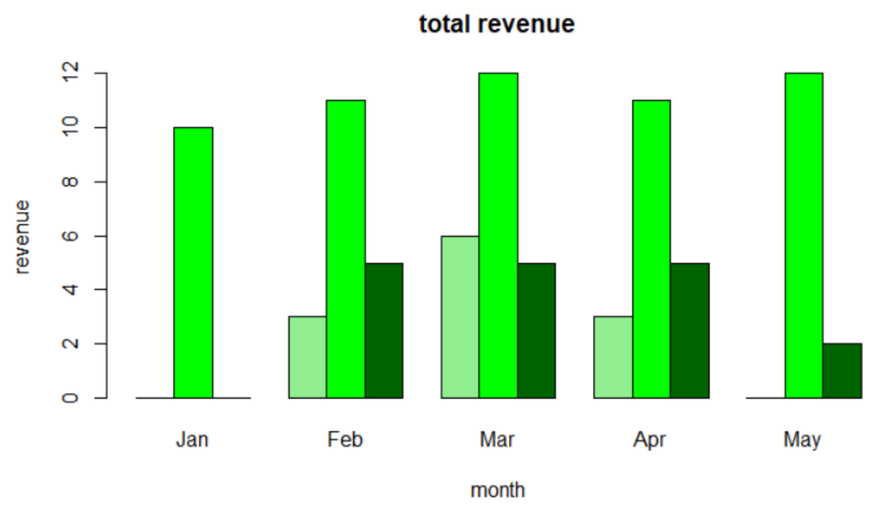
```
legend("topleft", source, cex = 1.3, fill = colors)
```



You can move the legend around to where it fits the best. The "cex" command also can change the size. Experiment with it so that it doesn't cover up the data.

The clustered version is obtained from the "beside" option.

```
barplot(Values, main = "total revenue", names.arg = months, xlab = "month", ylab = "rev
enue", col = colors, beside=TRUE)
```

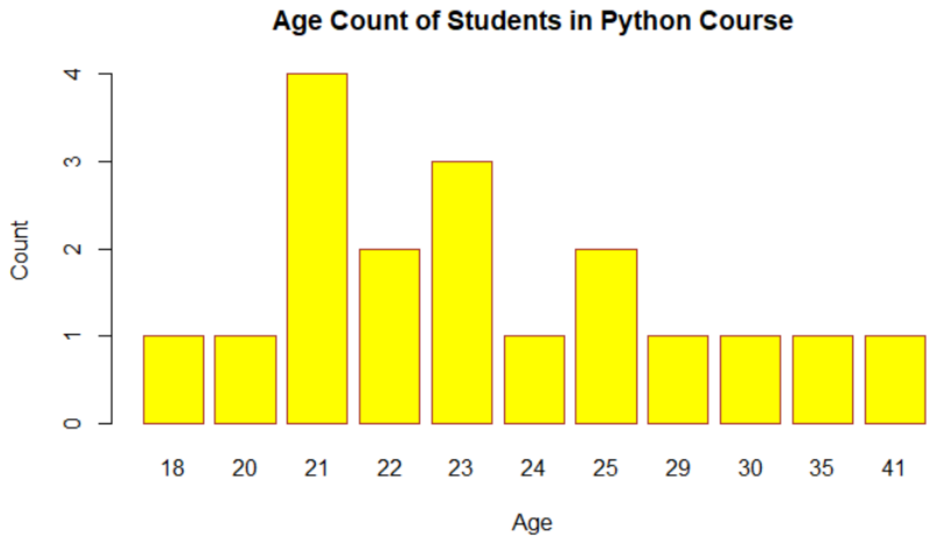


If you have some types of data (such as discrete data), you can summarize the data in R, and then build the graph from that. Consider the following age data.

```
age=c(18,20,21,21,21,21,22,23,23,23,22,24,25,25,29,30,35,41)
table(age)
age
18 20 21 22 23 24 25 29 30 35 41
 1  1  4  2  3  1  2  1  1  1  1
```

The table function can sort it into a table that we can then graph.

```
barplot(table(age),main="Age Count of Students in Python Course", xlab="Age", ylab="C
ount", border="brown", col="yellow")
```



For this data, there are gaps in the ages. Those won't appear in the graph unless you fill them in manually. But this works well for categorical data.

Practice.
I've included one additional example below, but feel free to experiment with your own data.

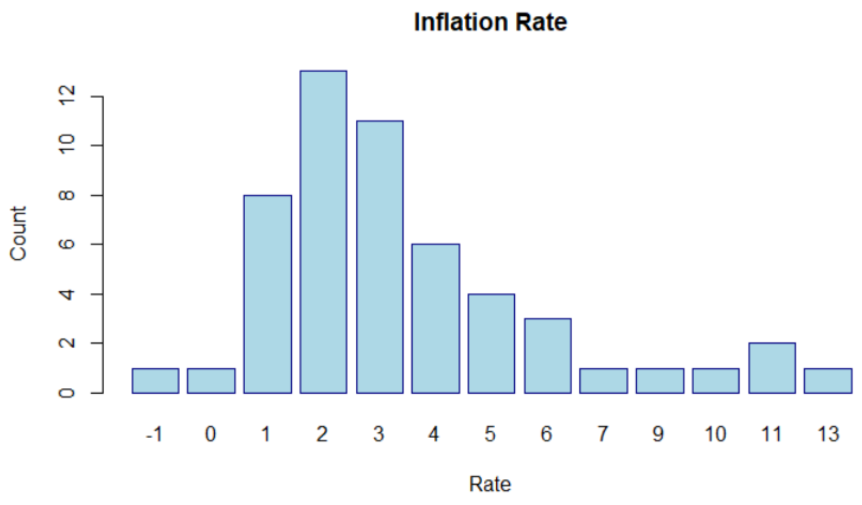
Round the data below to the floor value to bin it, and then make a bar graph or histogram of it.

```
InflationRate=c(1.59, 3.01, 2.78, 4.27, 5.46, 5.84, 4.30, 3.27, 6.16, 11.03, 9.20, 5.75, 6.50,
7.62, 11.22, 13.58, 10.35, 6.16, 3.22, 4.30, 3.55, 1.91, 3.66, 4.08, 4.83, 5.39, 4.25, 3.03, 2
.96, 2.61, 2.81, 2.93, 2.34, 1.55, 2.19, 3.38, 2.83, 1.59, 2.27, 2.68, 3.39, 3.24, 2.85, 3.85, -
0.34, 1.64, 3.16, 2.07, 1.47, 1.62, 0.12, 1.26, 2.13)
```

Solutions.
For the set of data above, use the floor function, and then graph it.

```
IRfloor=floor(InflationRate)

barplot(table(IRfloor),main="Inflation Rate", xlab="Rate", ylab="Count", border="navy",
col="lightblue")
```



Compare this result to the histogram results from another tutorial.