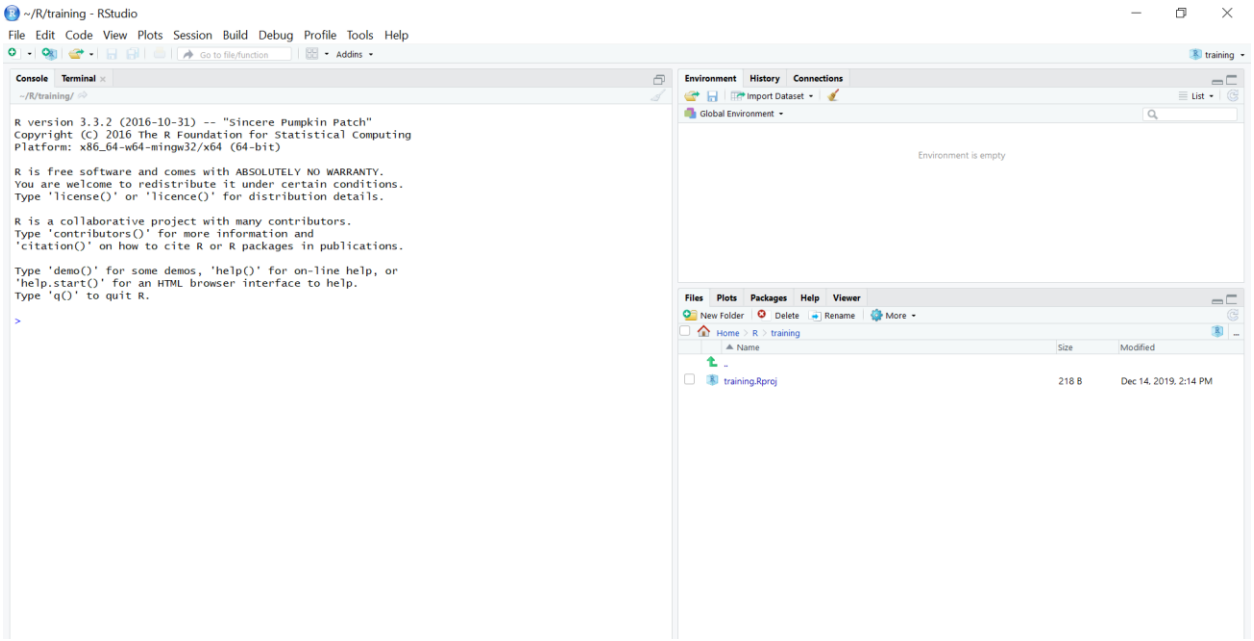
Mini-Tutorials for Creating Graphs in R

Histogram

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.



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 create a simple histogram from raw data. We'll add features as we go.

Copy the commands shown into the command line.

Step 1. Enter the data to be plotted in vector form.

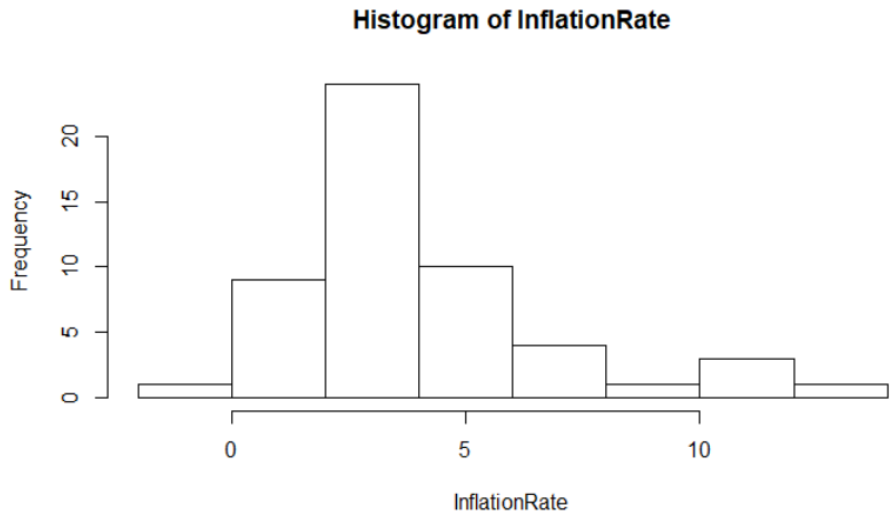
```
InflationRate=c(1.59, 3.01, 2.78, 4.27, 5.46, 5.84, 4.30, 3.27, 6.16, 11.03, 9.20, 5.75, 6.50,
7.62, 11.22, 13.58, 10.35, 6.16, 3.22, 4.30, 3.55, 1.91, 3.66, 4.08, 4.83, 5.39, 4.25, 3.03, 2
```

.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)

Histograms are quite simple to create in R with many default features built in that we can modify or not.

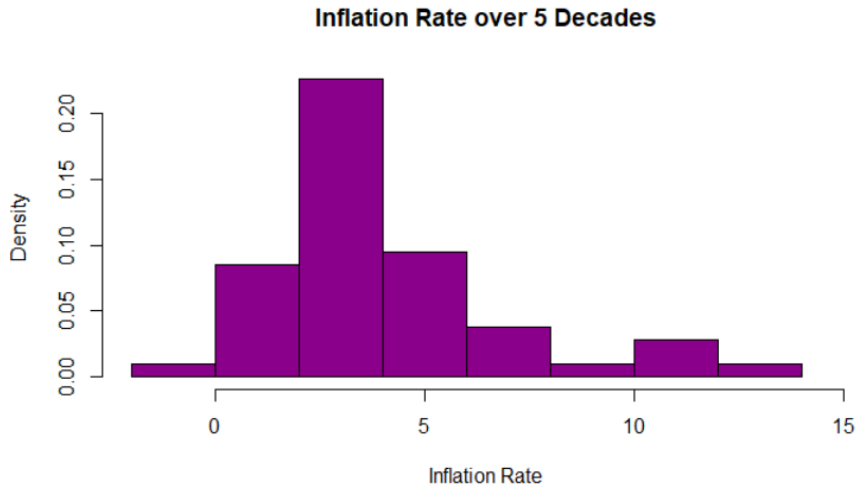
Step 2. Graphing the data.

```
hist(InflationRate)
```



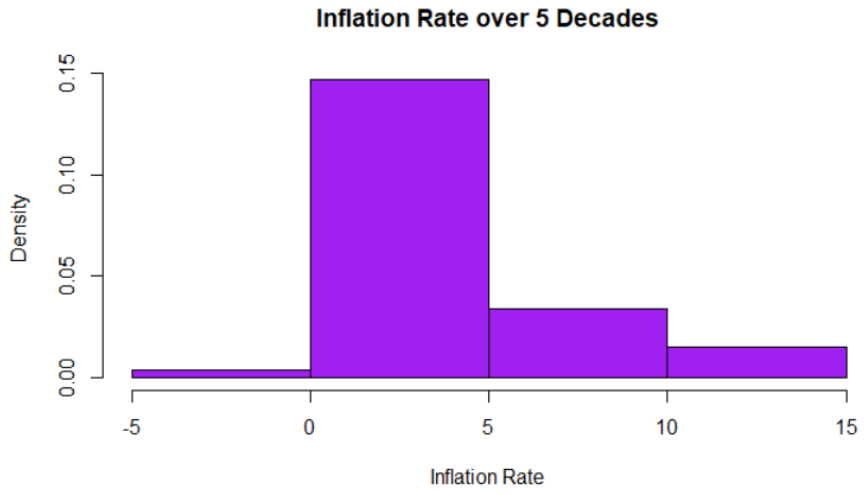
The graph we get is shown. The default option displays frequencies. You can change this to percentages. You can also adjust the axis range.

```
hist(InflationRate, main="Inflation Rate over 5 Decades", xlab="Inflation Rate", xlim=c(-2,15), col="darkmagenta",freq=FALSE)
```

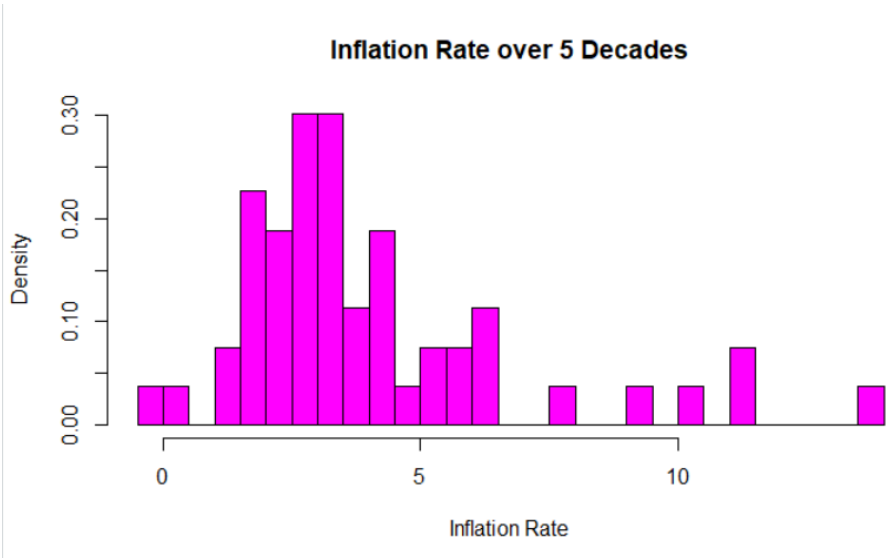


For this situation, R has defaulted to 8 bins (or 7 breaks), but we can change the number of bins by adding a "break" command. This will adjust the number of bins up or down. Let's look at 4 (3 breaks) bins, vs. 26 (25 breaks) bins for this same data.

```
hist(InflationRate, main="Inflation Rate over 5 Decades", xlab="Inflation Rate", col="purple", breaks=3, freq=FALSE)
```



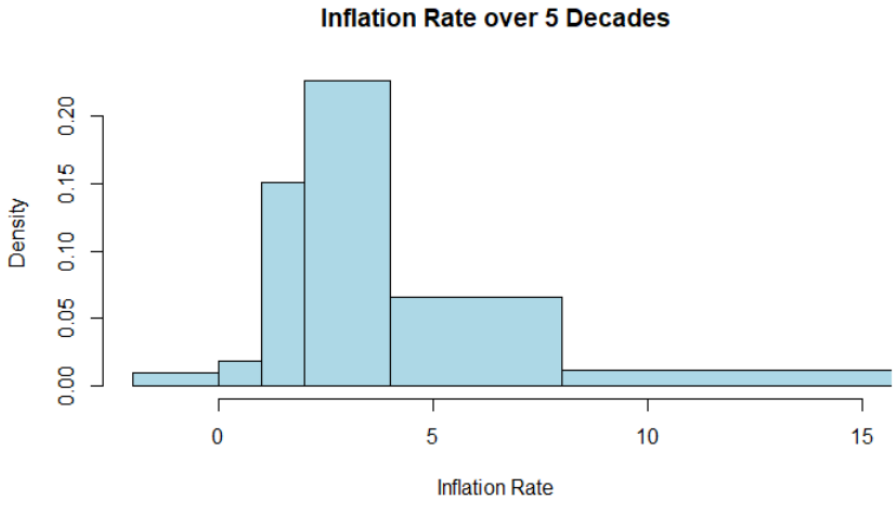

```
hist(InflationRate, main="Inflation Rate over 5 Decades", xlab="Inflation Rate", col="magenta", breaks=25, freq=FALSE)
```



Too few bins and you don't get much structure. Too many bins and you get too much detail. The default number is usually pretty good for common distributions. If you are going to adjust it, generally stay close to the default number based on the number of values in the data.

It's possible to create bins with non-uniform widths, though for statistical reasons this isn't recommended. To do this, your break command should include a vector that specifies the breaks including the beginning and end.

```
hist(InflationRate, main="Inflation Rate over 5 Decades", xlab="Inflation Rate", xlim=c(-2
,15), col="lightblue", breaks=c(-2,0,1,2,4,8,16), freq=FALSE)
```



Practice.
To get some random data to practice with, consider the following commands:

1. `test1=c(runif(100, min=80, max=100))`

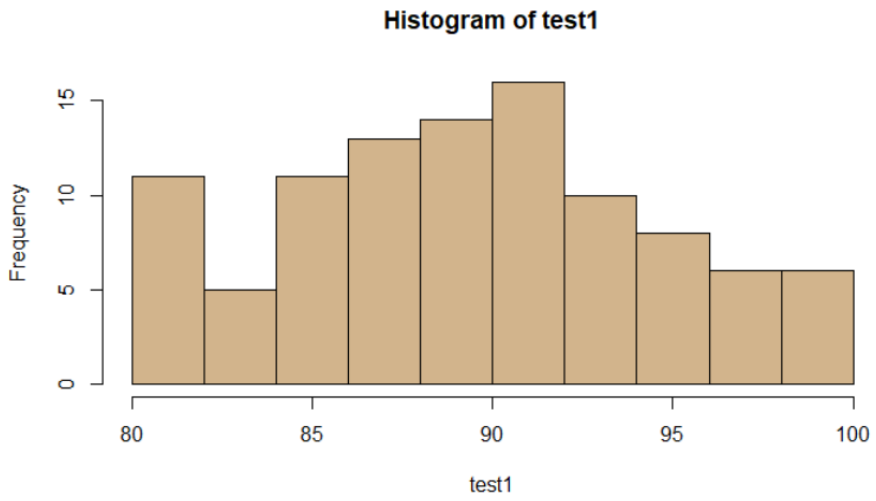
runif produces a list of uniformly distributed random numbers between the specified min and max values. In this case, 100 such numbers are generated.

2. `test2=c(rnorm(150, mean=64, sd=3.1))`

rnorm generates a list of normally distributed numbers with the specified mean and standard deviation. In this case 150 such numbers are generated.

Solutions.
Some example plots.

```
hist(test1, col="tan")
```




```
hist(test2, col="turquoise")
```