Basic Graphs

Statistics 427: R Programming

Module 4

2020

Basic Graphs

One thing I always tell students is that if you do not look at your data, as in a picture (graph), you do not have a full understanding of your data, regardless how much number crunching you do. Most often in statistics, in order to choose the most appropriate model, you must know what the distribution of the data looks like; that is *look at a graph*. Graphs are crucial to data science, *all* sciences (all disciplines); even a small data table is nearly incomprehensible by itself. They allow scientists to visualize quantitative patterns.

Iris data example I

Edgar Anderson's Iris Data

This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*.

This dataset is a built-in dataset for our use; we use the `data()` function.

Iris Dataset

```
data(iris)
iris
```

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
7            4.6         3.4          1.4         0.3     setosa
8            5.0         3.4          1.5         0.2     setosa
9            4.4         2.9          1.4         0.2     setosa
10           4.9         3.1          1.5         0.1     setosa
11           5.4         3.7          1.5         0.2     setosa
12           4.8         3.4          1.6         0.2     setosa
13           4.8         3.0          1.4         0.1     setosa
14           4.3         3.0          1.1         0.1     setosa
15           5.8         4.0          1.2         0.2     setosa
16           5.7         4.4          1.5         0.4     setosa
17           5.4         3.9          1.3         0.4     setosa
18           5.1         3.5          1.4         0.3     setosa
```

19	5.7	3.8	1.7	0.3	setosa
20	5.1	3.8	1.5	0.3	setosa
21	5.4	3.4	1.7	0.2	setosa
22	5.1	3.7	1.5	0.4	setosa
23	4.6	3.6	1.0	0.2	setosa
24	5.1	3.3	1.7	0.5	setosa
25	4.8	3.4	1.9	0.2	setosa
26	5.0	3.0	1.6	0.2	setosa
27	5.0	3.4	1.6	0.4	setosa
28	5.2	3.5	1.5	0.2	setosa
29	5.2	3.4	1.4	0.2	setosa
30	4.7	3.2	1.6	0.2	setosa
31	4.8	3.1	1.6	0.2	setosa
32	5.4	3.4	1.5	0.4	setosa
33	5.2	4.1	1.5	0.1	setosa
34	5.5	4.2	1.4	0.2	setosa
35	4.9	3.1	1.5	0.2	setosa
36	5.0	3.2	1.2	0.2	setosa
37	5.5	3.5	1.3	0.2	setosa
38	4.9	3.6	1.4	0.1	setosa
39	4.4	3.0	1.3	0.2	setosa
40	5.1	3.4	1.5	0.2	setosa
41	5.0	3.5	1.3	0.3	setosa
42	4.5	2.3	1.3	0.3	setosa
43	4.4	3.2	1.3	0.2	setosa
44	5.0	3.5	1.6	0.6	setosa
45	5.1	3.8	1.9	0.4	setosa
46	4.8	3.0	1.4	0.3	setosa
47	5.1	3.8	1.6	0.2	setosa
48	4.6	3.2	1.4	0.2	setosa
49	5.3	3.7	1.5	0.2	setosa
50	5.0	3.3	1.4	0.2	setosa
51	7.0	3.2	4.7	1.4	versicolor
52	6.4	3.2	4.5	1.5	versicolor
53	6.9	3.1	4.9	1.5	versicolor
54	5.5	2.3	4.0	1.3	versicolor
55	6.5	2.8	4.6	1.5	versicolor
56	5.7	2.8	4.5	1.3	versicolor
57	6.3	3.3	4.7	1.6	versicolor
58	4.9	2.4	3.3	1.0	versicolor
59	6.6	2.9	4.6	1.3	versicolor
60	5.2	2.7	3.9	1.4	versicolor
61	5.0	2.0	3.5	1.0	versicolor
62	5.9	3.0	4.2	1.5	versicolor
63	6.0	2.2	4.0	1.0	versicolor
64	6.1	2.9	4.7	1.4	versicolor
65	5.6	2.9	3.6	1.3	versicolor
66	6.7	3.1	4.4	1.4	versicolor
67	5.6	3.0	4.5	1.5	versicolor
68	5.8	2.7	4.1	1.0	versicolor
69	6.2	2.2	4.5	1.5	versicolor
70	5.6	2.5	3.9	1.1	versicolor
71	5.9	3.2	4.8	1.8	versicolor
72	6.1	2.8	4.0	1.3	versicolor

73	6.3	2.5	4.9	1.5	versicolor
74	6.1	2.8	4.7	1.2	versicolor
75	6.4	2.9	4.3	1.3	versicolor
76	6.6	3.0	4.4	1.4	versicolor
77	6.8	2.8	4.8	1.4	versicolor
78	6.7	3.0	5.0	1.7	versicolor
79	6.0	2.9	4.5	1.5	versicolor
80	5.7	2.6	3.5	1.0	versicolor
81	5.5	2.4	3.8	1.1	versicolor
82	5.5	2.4	3.7	1.0	versicolor
83	5.8	2.7	3.9	1.2	versicolor
84	6.0	2.7	5.1	1.6	versicolor
85	5.4	3.0	4.5	1.5	versicolor
86	6.0	3.4	4.5	1.6	versicolor
87	6.7	3.1	4.7	1.5	versicolor
88	6.3	2.3	4.4	1.3	versicolor
89	5.6	3.0	4.1	1.3	versicolor
90	5.5	2.5	4.0	1.3	versicolor
91	5.5	2.6	4.4	1.2	versicolor
92	6.1	3.0	4.6	1.4	versicolor
93	5.8	2.6	4.0	1.2	versicolor
94	5.0	2.3	3.3	1.0	versicolor
95	5.6	2.7	4.2	1.3	versicolor
96	5.7	3.0	4.2	1.2	versicolor
97	5.7	2.9	4.2	1.3	versicolor
98	6.2	2.9	4.3	1.3	versicolor
99	5.1	2.5	3.0	1.1	versicolor
100	5.7	2.8	4.1	1.3	versicolor
101	6.3	3.3	6.0	2.5	virginica
102	5.8	2.7	5.1	1.9	virginica
103	7.1	3.0	5.9	2.1	virginica
104	6.3	2.9	5.6	1.8	virginica
105	6.5	3.0	5.8	2.2	virginica
106	7.6	3.0	6.6	2.1	virginica
107	4.9	2.5	4.5	1.7	virginica
108	7.3	2.9	6.3	1.8	virginica
109	6.7	2.5	5.8	1.8	virginica
110	7.2	3.6	6.1	2.5	virginica
111	6.5	3.2	5.1	2.0	virginica
112	6.4	2.7	5.3	1.9	virginica
113	6.8	3.0	5.5	2.1	virginica
114	5.7	2.5	5.0	2.0	virginica
115	5.8	2.8	5.1	2.4	virginica
116	6.4	3.2	5.3	2.3	virginica
117	6.5	3.0	5.5	1.8	virginica
118	7.7	3.8	6.7	2.2	virginica
119	7.7	2.6	6.9	2.3	virginica
120	6.0	2.2	5.0	1.5	virginica
121	6.9	3.2	5.7	2.3	virginica
122	5.6	2.8	4.9	2.0	virginica
123	7.7	2.8	6.7	2.0	virginica
124	6.3	2.7	4.9	1.8	virginica
125	6.7	3.3	5.7	2.1	virginica
126	7.2	3.2	6.0	1.8	virginica

```
127          6.2         2.8          4.8         1.8  virginica
128          6.1         3.0          4.9         1.8  virginica
129          6.4         2.8          5.6         2.1  virginica
130          7.2         3.0          5.8         1.6  virginica
131          7.4         2.8          6.1         1.9  virginica
132          7.9         3.8          6.4         2.0  virginica
133          6.4         2.8          5.6         2.2  virginica
134          6.3         2.8          5.1         1.5  virginica
135          6.1         2.6          5.6         1.4  virginica
136          7.7         3.0          6.1         2.3  virginica
137          6.3         3.4          5.6         2.4  virginica
138          6.4         3.1          5.5         1.8  virginica
139          6.0         3.0          4.8         1.8  virginica
140          6.9         3.1          5.4         2.1  virginica
141          6.7         3.1          5.6         2.4  virginica
142          6.9         3.1          5.1         2.3  virginica
143          5.8         2.7          5.1         1.9  virginica
144          6.8         3.2          5.9         2.3  virginica
145          6.7         3.3          5.7         2.5  virginica
146          6.7         3.0          5.2         2.3  virginica
147          6.3         2.5          5.0         1.9  virginica
148          6.5         3.0          5.2         2.0  virginica
149          6.2         3.4          5.4         2.3  virginica
150          5.9         3.0          5.1         1.8  virginica
```

Datasets vs. Vectors

A dataset is an object that has objects (variables/vectors) within it. To access the variables for arguments in a dataset (either with `data()` or read in), you have to state the object name, followed by a dollar sign (`$`), followed by the variable name; this is called a two-level name.

`dataset$variable1`

The reason to bring this up now is for this example (or the other option was to type a lot to create individual vectors). Here I will show one way to deal with variable names **but this is not efficient**. There are *better* methods we will learn in the next module.

Iris Dataset II

To look at the individual data points, we can call the variable and it will show in a list. For the moment, we have to call it using the two-level name. `iris$Sepal.Length`

```
iris$Sepal.Length
```

```
  [1] 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 4.3 5.8 5.7 5.4 5.1
 [19] 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0
 [37] 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5
 [55] 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1
 [73] 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3 5.6 5.5
 [91] 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3
[109] 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0 6.9 5.6 7.7 6.3 6.7 7.2
[127] 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8
[145] 6.7 6.7 6.3 6.5 6.2 5.9
```

Iris Dataset III

Sepal length is numeric, as are sepal width, petal length, and petal width. Species has no numbers in the variable at all, it is qualitative (categorical, character, etc.). Sometimes the names will be in quotes (depends on what kind of character data it is... more on that later). Based on datatypes, we can figure out which graphs are appropriate given the data type.

```
iris$Species
```

```
  [1] setosa     setosa     setosa     setosa     setosa     setosa
  [7] setosa     setosa     setosa     setosa     setosa     setosa
 [13] setosa     setosa     setosa     setosa     setosa     setosa
 [19] setosa     setosa     setosa     setosa     setosa     setosa
 [25] setosa     setosa     setosa     setosa     setosa     setosa
 [31] setosa     setosa     setosa     setosa     setosa     setosa
 [37] setosa     setosa     setosa     setosa     setosa     setosa
 [43] setosa     setosa     setosa     setosa     setosa     setosa
 [49] setosa     setosa     versicolor versicolor versicolor versicolor
 [55] versicolor versicolor versicolor versicolor versicolor versicolor
 [61] versicolor versicolor versicolor versicolor versicolor versicolor
 [67] versicolor versicolor versicolor versicolor versicolor versicolor
 [73] versicolor versicolor versicolor versicolor versicolor versicolor
 [79] versicolor versicolor versicolor versicolor versicolor versicolor
 [85] versicolor versicolor versicolor versicolor versicolor versicolor
 [91] versicolor versicolor versicolor versicolor versicolor versicolor
 [97] versicolor versicolor versicolor versicolor virginica  virginica
[103] virginica  virginica  virginica  virginica  virginica  virginica
[109] virginica  virginica  virginica  virginica  virginica  virginica
[115] virginica  virginica  virginica  virginica  virginica  virginica
[121] virginica  virginica  virginica  virginica  virginica  virginica
[127] virginica  virginica  virginica  virginica  virginica  virginica
[133] virginica  virginica  virginica  virginica  virginica  virginica
[139] virginica  virginica  virginica  virginica  virginica  virginica
[145] virginica  virginica  virginica  virginica  virginica  virginica
Levels: setosa versicolor virginica
```

Iris Dataset IV

The following example is just for this one time to help with the two-level name of the variables for ease of coding. There are better ways to deal with it than this method we will learn later.

```
sl=iris$Sepal.Length
sw=iris$Sepal.Width
pl=iris$Petal.Length
pw=iris$Petal.Width
species=iris$Species
```

Graphs of One Variable

Stripchart: A striptchart (also called a dotplot) is a simple visualization of the data; use with quantitative data

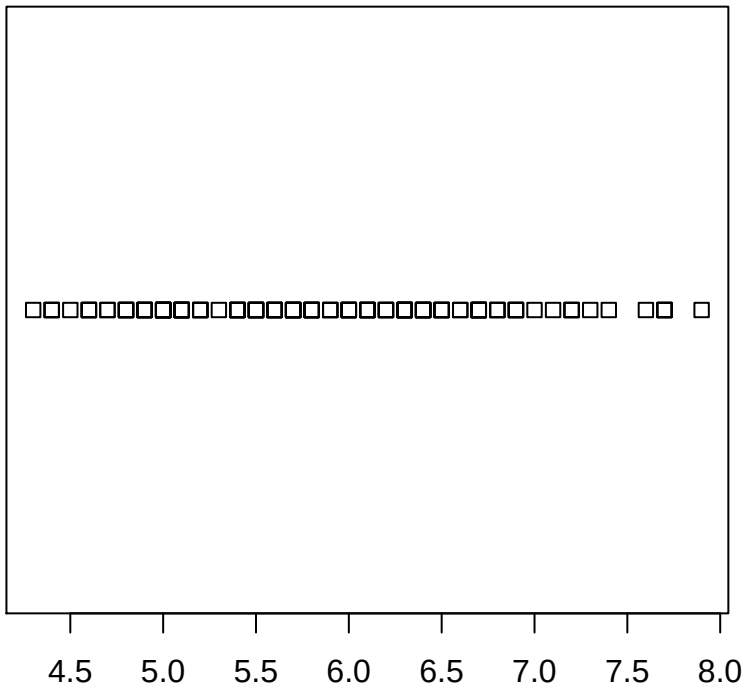
General form:

```
stripchart(x,main="Title",xlab="X label",ylab="Y label",method="",pch= , cex= )
```

`x`: vector of values, can also be a formula (more later)
`main=" "`: user-defined title; default is no title
`xlab`, `ylab`: x- and y-axes labels; default is no labels
`method`: method to be used to separate coincident points. Default "`overplot`" causes such points to be overplotted, "`jitter`" jitters the points, or "`stack`" have coincident points stacked
`pch`, `cex`: graphics customization; pch is symbol type, cex is size of the symbol
`...`: more options, soooo many more options (type ?stripchart in console for help)

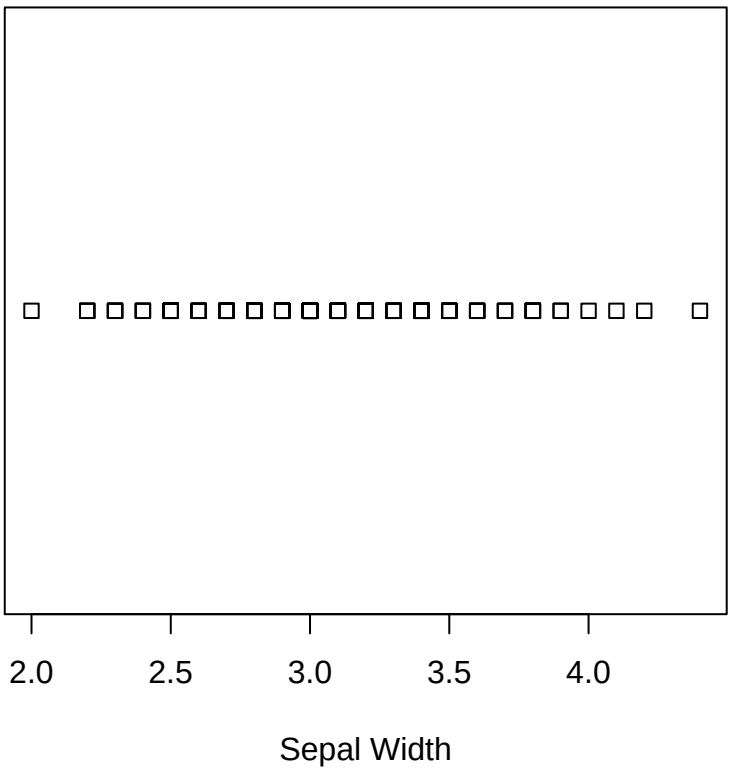
Stripchart Sepal Length

```
stripchart(sl)
```



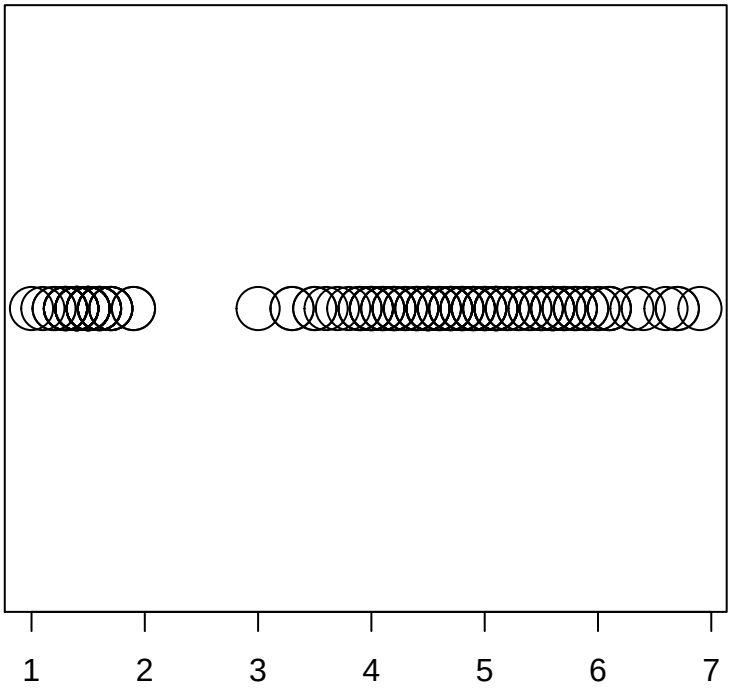
Stripchart Sepal Width

```
stripchart(sw,main="Iris Sepal Width",xlab="Sepal Width")
```



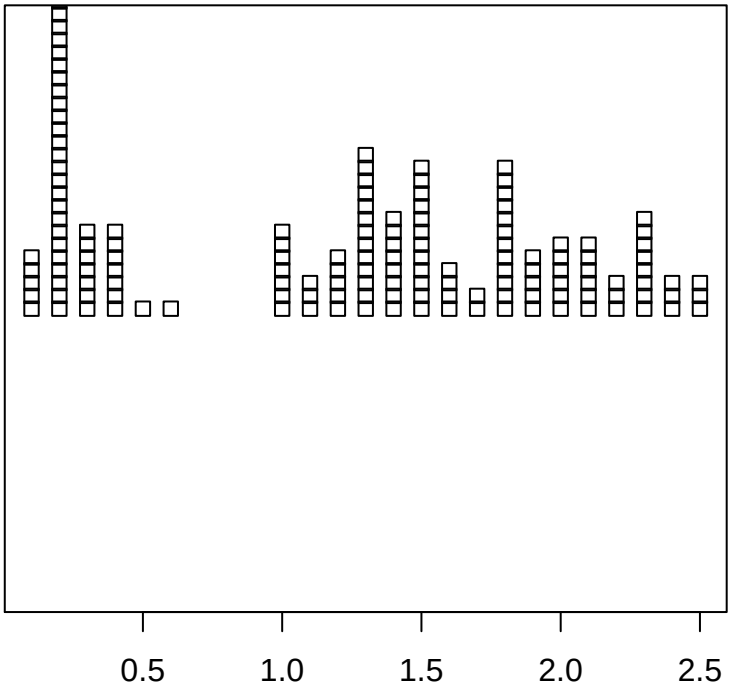
Stripchart Petal Length

```
stripchart(pl,pch=1,cex=3)
```



Stripchart Petal Width

```
stripchart(pw,method="stack")
```



Histograms

A histogram represents the frequency (or sometimes relative frequency/density/percents) of the data points in an interval as a rectangle over the interval, with the area of the rectangle equal to the frequency. The histogram function in `R` usually does a decent job at creating the classes (bins) for the data on its own. There is a way to control more of what the graphs can do with more extensive options, but we are still looking at basic graphs.

General Form of `hist()`

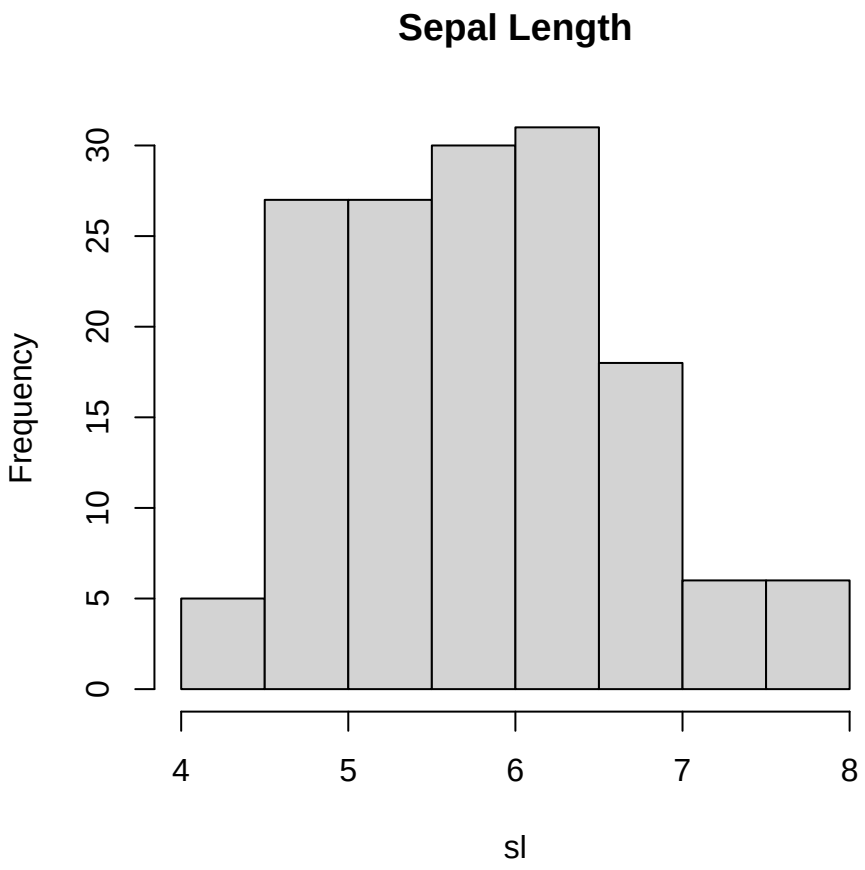
Histogram function:

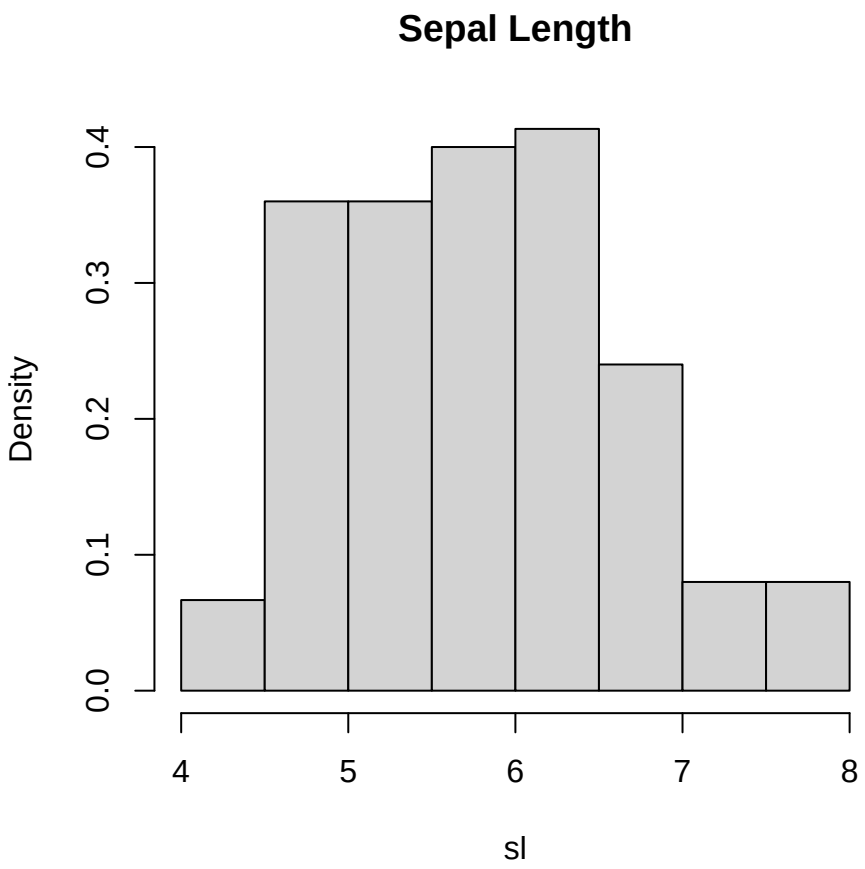
```
hist(x,freq= ,main=" ",xlab=" ",ylab=" ",xlim=c(),ylim=c(),...)
```

`x`: vector of values
`freq`: T (TRUE) is the default for frequencies, F (FALSE) for relative frequency plot
`main`, `xlab`, `ylab`: title, x-, and y-axes labels
`xlim`, `ylim`: the range of x and y values for axes
`...`: more options

Histogram of Iris Sepal Length

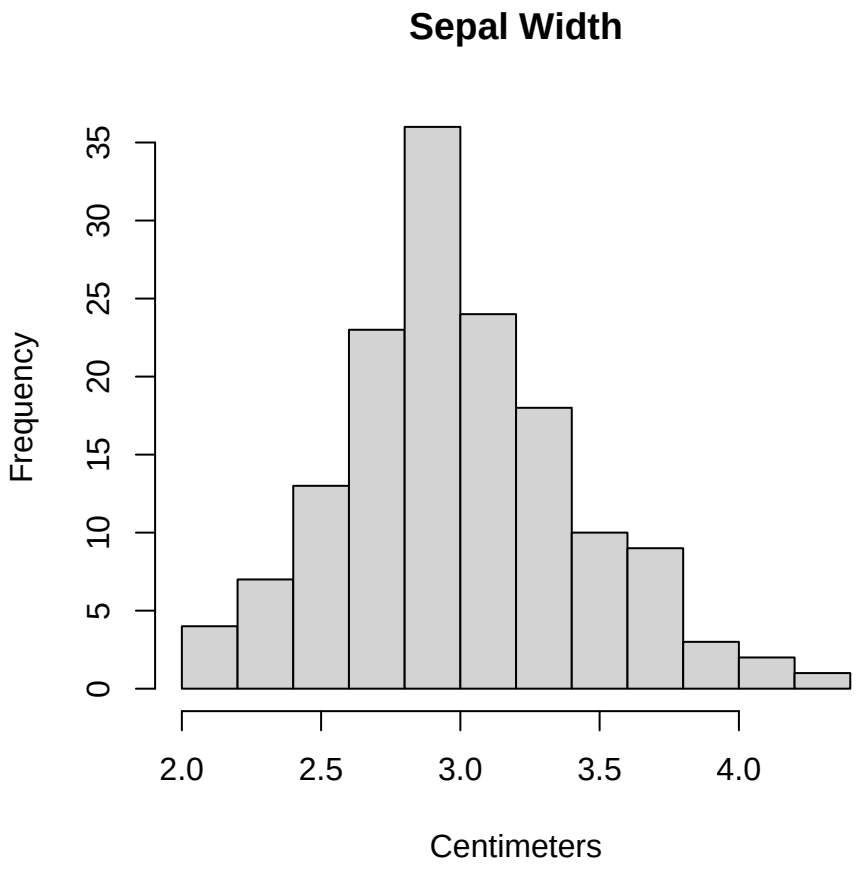
```r
hist(sl,main='Sepal Length'); hist(sl,freq=F,main='Sepal Length')
```





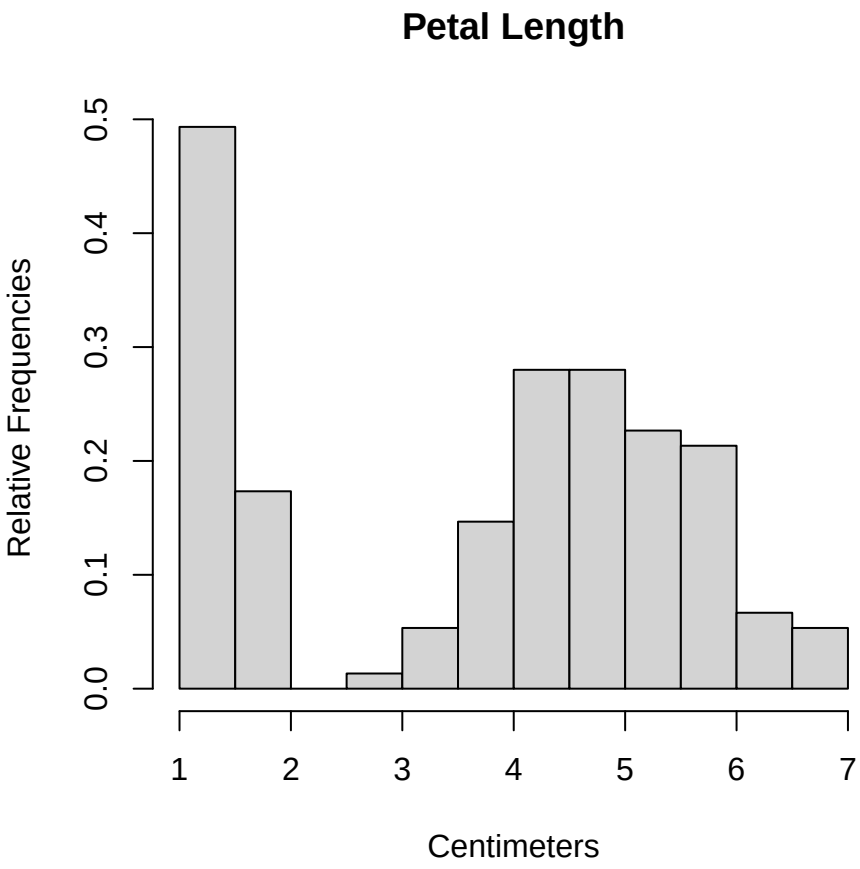
Histogram of Iris Sepal Width

```
hist(sw,main='Sepal Width',xlab='Centimeters')
```



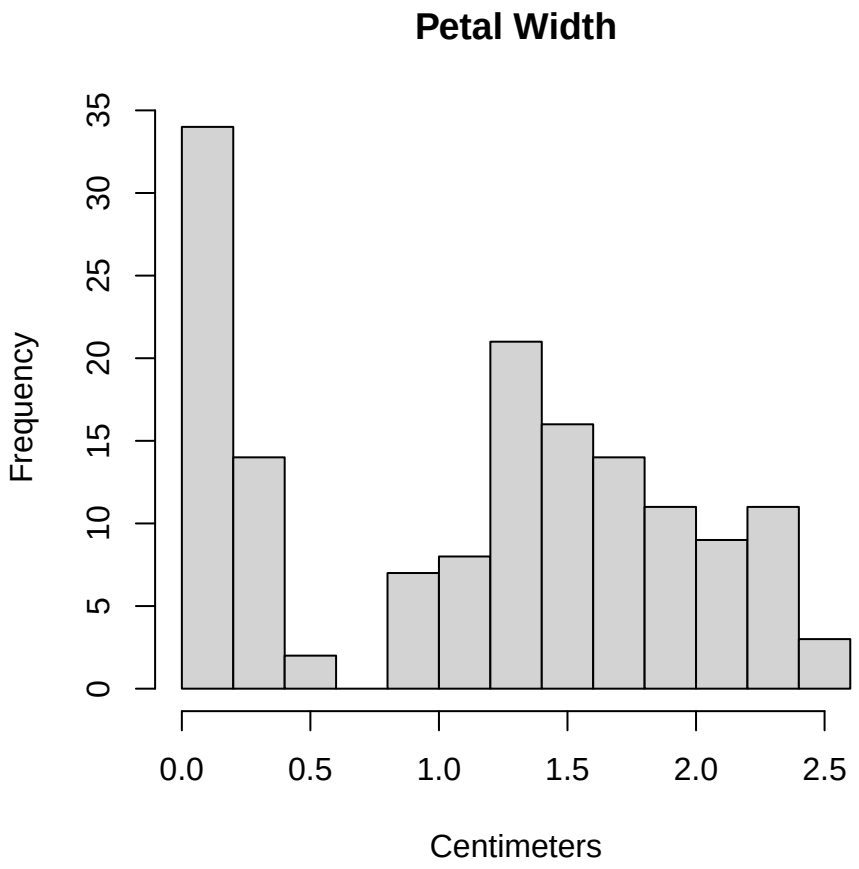
Histogram of Iris Petal Length

```
hist(pl,freq=F,main='Petal Length',xlab='Centimeters',ylab='Relative Frequencies')
```



Histogram of Iris Petal Width

```
hist(pw,main='Petal Width',xlab='Centimeters')
```



Stemplot (Stem-and-Leaf)

Sometimes the dataset is small and a histogram is not really appropriate because of that. It displays all of the data points, which is why it is best for small datasets (but can be used with large datasets as well). It is a typographic construct that allows that viewer to reconstruct all of the original data, to at least two digits. Each data point has a stem and a leaf. The stem is the number in the left column of the plot, and the leaves are the numbers extending horizontally to the right.

General Form of `stem()`

`stem(x,...)`

`x`: vector of values
`...`: more options

Stemplot of Sepal Length

```
stem(sl)
```

```
  The decimal point is 1 digit(s) to the left of the |

  42 | 0
  44 | 0000
  46 | 000000
  48 | 00000000000
  50 | 0000000000000000000
```

```
  52 | 00000
  54 | 0000000000000
  56 | 00000000000000
  58 | 0000000000
  60 | 000000000000
  62 | 0000000000000
  64 | 000000000000
  66 | 0000000000
  68 | 0000000
  70 | 00
  72 | 0000
  74 | 0
  76 | 00000
  78 | 0
```

Stemplot of Sepal Width

```
stem(sw)
```

```
  The decimal point is 1 digit(s) to the left of the |

  20 | 0
  21 |
  22 | 000
  23 | 0000
  24 | 000
  25 | 00000000
  26 | 00000
  27 | 000000000
  28 | 00000000000000
  29 | 0000000000
  30 | 00000000000000000000000000
  31 | 00000000000
  32 | 0000000000000
  33 | 000000
  34 | 000000000000
  35 | 000000
  36 | 0000
  37 | 000
  38 | 000000
  39 | 00
  40 | 0
  41 | 0
  42 | 0
  43 |
  44 | 0
```

Stemplot of Petal Length

```
stem(pl)
```

```
  The decimal point is at the |

  1 | 01223333333444444444444
  1 | 55555555555556666666777799
  2 |
  2 |
  3 | 033
  3 | 55678999
  4 | 000001112222334444
  4 | 55555555666777778888999999
  5 | 000011111111223344
  5 | 55566666677788899
  6 | 0011134
  6 | 6779
```

Stemplot of Petal Width

```
stem(pw)
```

```
  The decimal point is 1 digit(s) to the left of the |

   1 | 00000
   2 | 00000000000000000000000000000
   3 | 0000000
   4 | 0000000
   5 | 0
   6 | 0
   7 |
   8 |
   9 |
  10 | 0000000
  11 | 000
  12 | 00000
  13 | 0000000000000
  14 | 00000000
  15 | 000000000000
  16 | 0000
  17 | 00
  18 | 000000000000
  19 | 00000
  20 | 000000
  21 | 000000
  22 | 000
  23 | 00000000
  24 | 000
  25 | 000
```

Boxplots (box-and-whisker plots)

A boxplot relies on five numbers to summarize all of the data in the variable. The "5 Number summary" consists of the minimum, quartile 1, median, quartile 3, and the maximum.

There are functions to find the values of the 5 number summary, including the individual commands and some that summarize multiple calculations.

5 number summary

minimum: (min) the smallest observation. `min(x)` where `x` is a vector of values

quartile 1: (Q1) the 25^{th} percentile; 25% of the data points are less than Q1 and 75% are greater than Q1. `quantile(x,probs=.25)` where `x` is a vector of values

median: aka 50^{th} percentile; the middle observation; 50% of the data points are less than the median and 50% are greater than the median. `median(x)` where `x` is a vector of values

quartile 3: (Q3) the 75^{th} percentile; 75% of the data points are less than Q3 and 25% are greater than Q3. `quantile(x,probs=.75)` where `x` is a vector of values

maximum: (max) the largest observation. `max(x)` where `x` is a vector of values

General form of `summary()`

One summary command, called `summary(x)`, where `x` is a vector of values or a data frame (a object like the `iris` dataset with multiple vectors). The `summary()` command gives, the 5 number summary, the mean, and sample sizes.

```
summary(sl)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.300   5.100   5.800   5.843   6.400   7.900
```

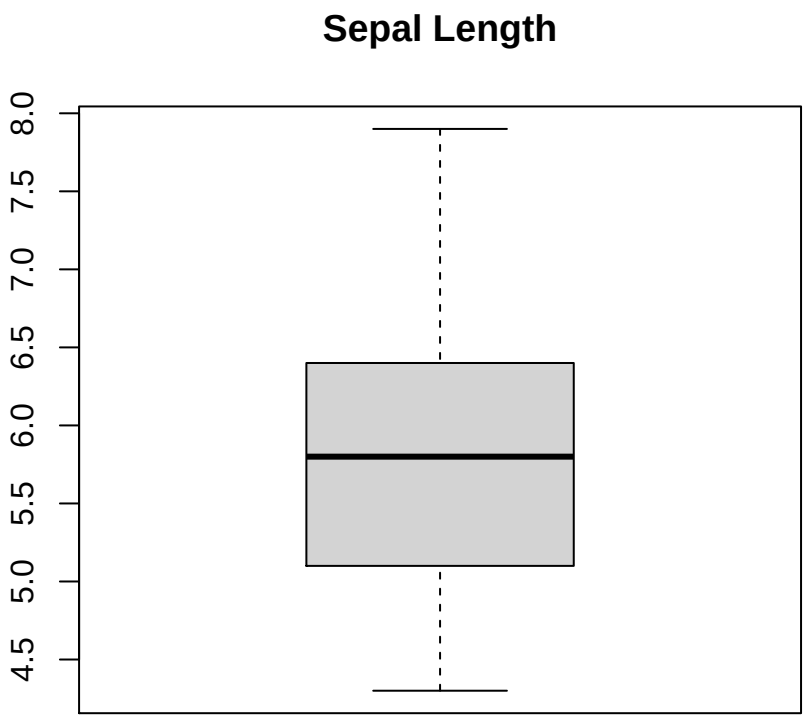
General Form of `boxplot()`

```
boxplot(x,main=" ",xlab=" ",ylab=" ",horizontal=F,xlim=c(),ylim=c(),...)
```

`x`: vector of values or a formula (`y~x`; `x` is quantitative, `y` is character)
`main`, `xlab`, `ylab`: title, x-, and y-axes labels
`xlim`, `ylim`: the range of x and y values for axes
`horizontal=F`: logical; F (FALSE) is the default, T (TRUE) is horizontal
`...`: more options

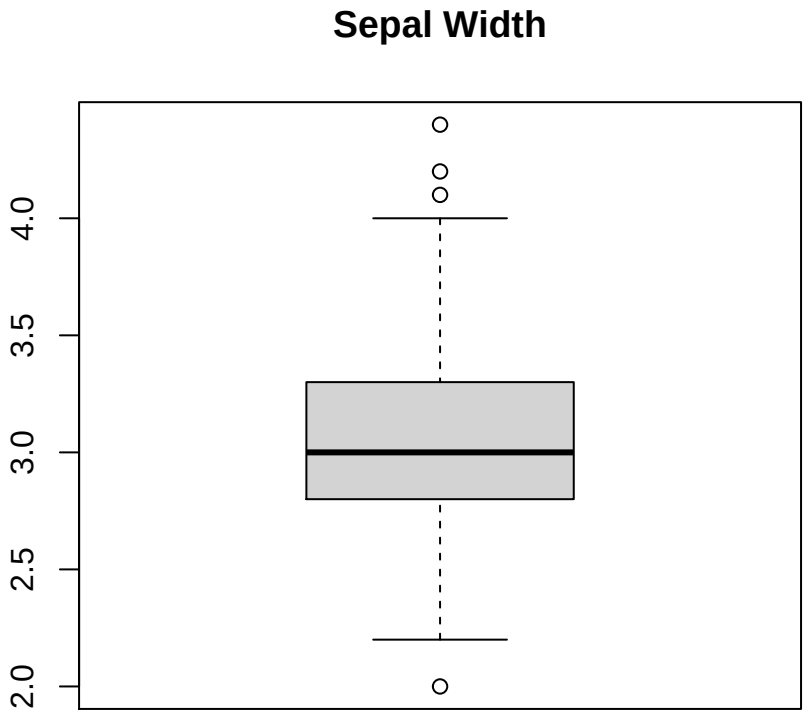
Boxplot of Sepal Length

```
boxplot(sl,main='Sepal Length')
```



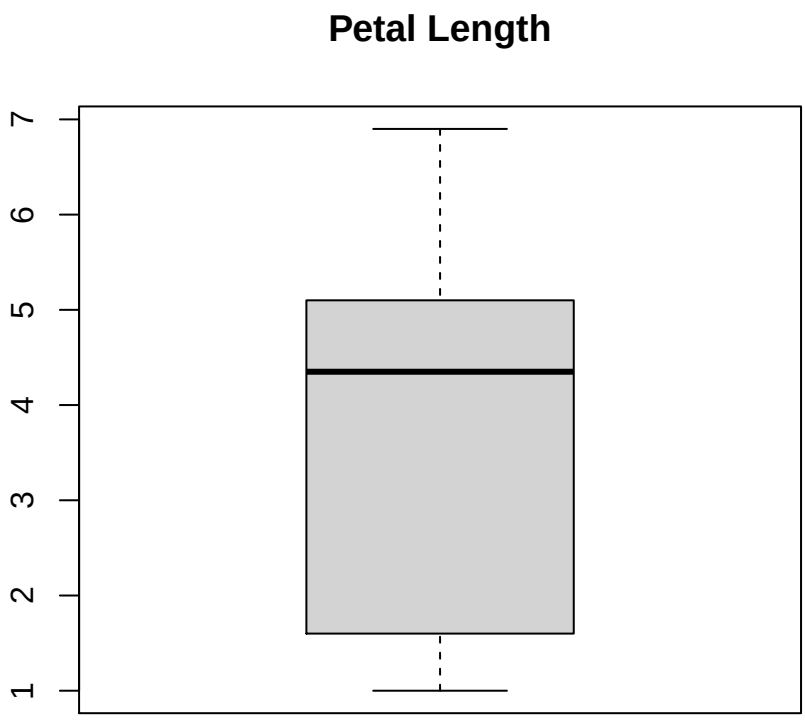
Boxplot of Sepal Width

```
boxplot(sw,main='Sepal Width')
```



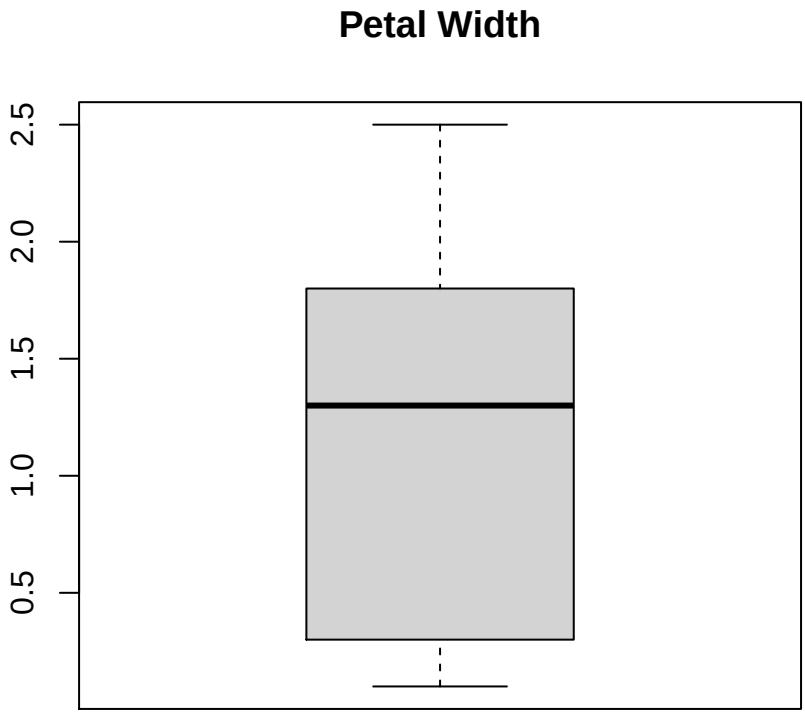
Boxplot of Petal Length

```
boxplot(pl,main='Petal Length')
```



Boxplot of Petal Width

```
boxplot(pw,main='Petal Width')
```



Graphs of two variables

Various types of graphs are helpful for investigating relationships between two variables.

Scatterplot: used when both variables are quantitative; it shows the values of two variables recorded from each subject/unit as an ordered pair on an x-y plot.

Side-by-side boxplots: graph the values by categorical group(s).

Bar charts: graph depicting frequencies (or percents) of a numeric variable by a categorical variable

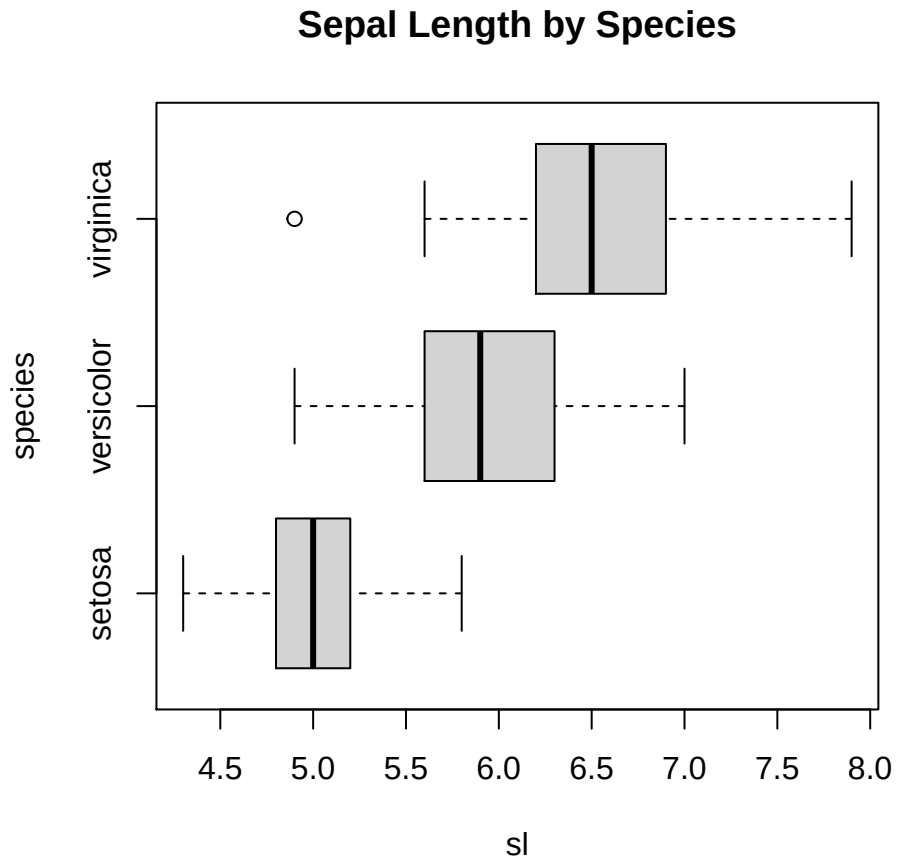
Pie charts: (are evil and you should never use them) graph depicting frequencies (or percents) of a numeric variable by a categorical variable

Boxplots by a Categorical Variable

That is where the `formula` part comes in. A formula, `y~x`, means "y as a function of x" where `y` is the numeric vector and `x` is the character vector, so split `y` by the groups of `x`. [Note: `x` has usually been a numeric vector but here it is the character vector]

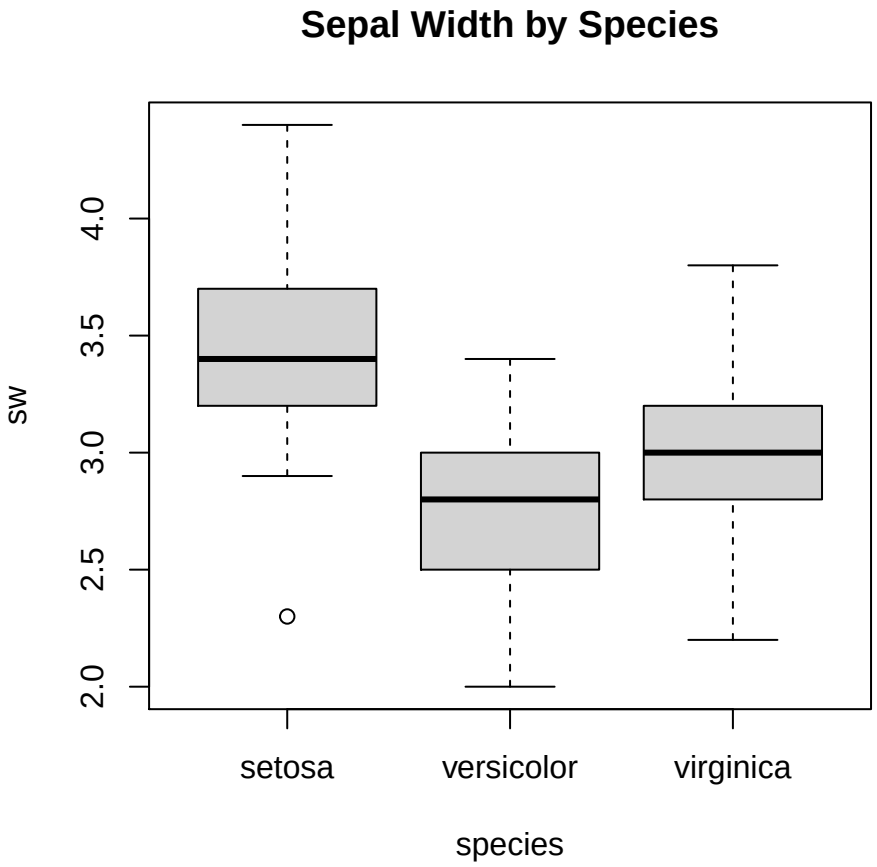
Boxplot Sepal Length by Species

```
boxplot(sl~species,main='Sepal Length by Species',horizontal=T)
```



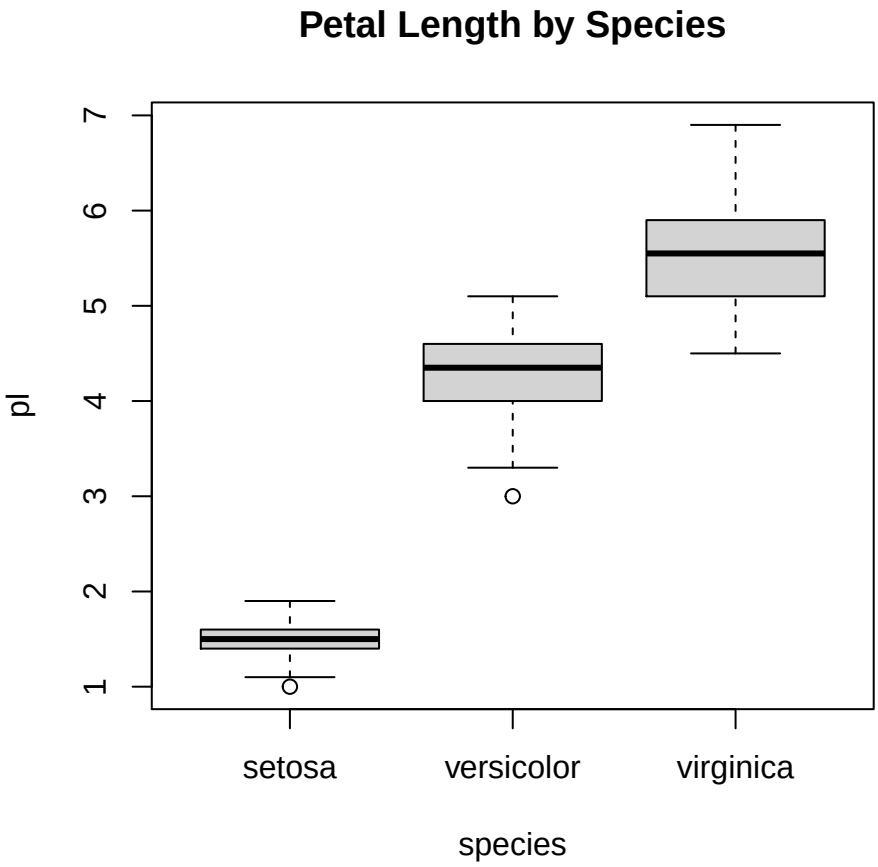
Boxplot Sepal Width by Species

```
boxplot(sw~species,main='Sepal Width by Species')
```



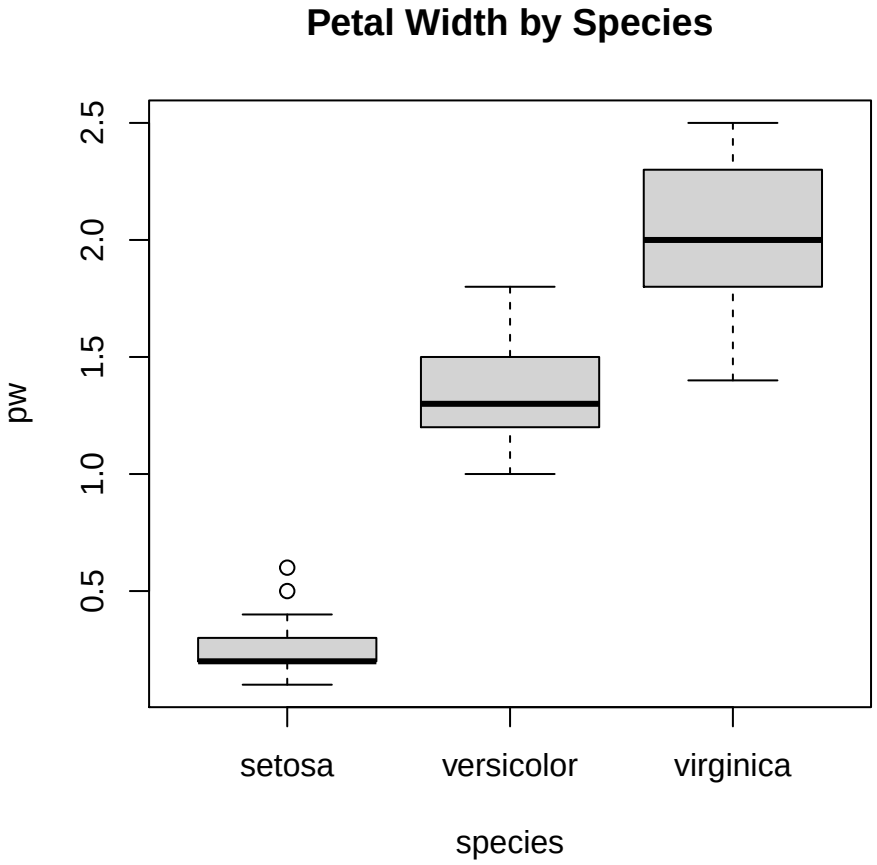
Boxplot Petal Length by Species

```
boxplot(pl~species,main='Petal Length by Species')
```



Boxplot Petal Width by Species

```
boxplot(pw~species,main='Petal Width by Species')
```



General form of `barplot()` and `pie()`

`barplot(x,horiz= ,...)`

> `x`: numeric vector or formula `y~x`; sometimes `x` needs to be a table
> `horiz`: F is the default, T gives horizontal plot
> `...`: more options

`pie(x,...)`

> `x` numeric vector (values displayed as areas of pie slices)
> `...`: more options

Creating a table of counts with `table()`

`table()` (similarly `as.table()` and `is.table()` both work to do the same thing) uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels (the categories).

`table(x,...)`

> `x`: a vector of data to categorize
> `...`: more options including main, xlab, etc. (type `?table` in console for help page)

Iris data tables

```
sptable=table(species) # counts of flower species
sptable
```

```
species
    setosa versicolor  virginica
        50         50         50
```

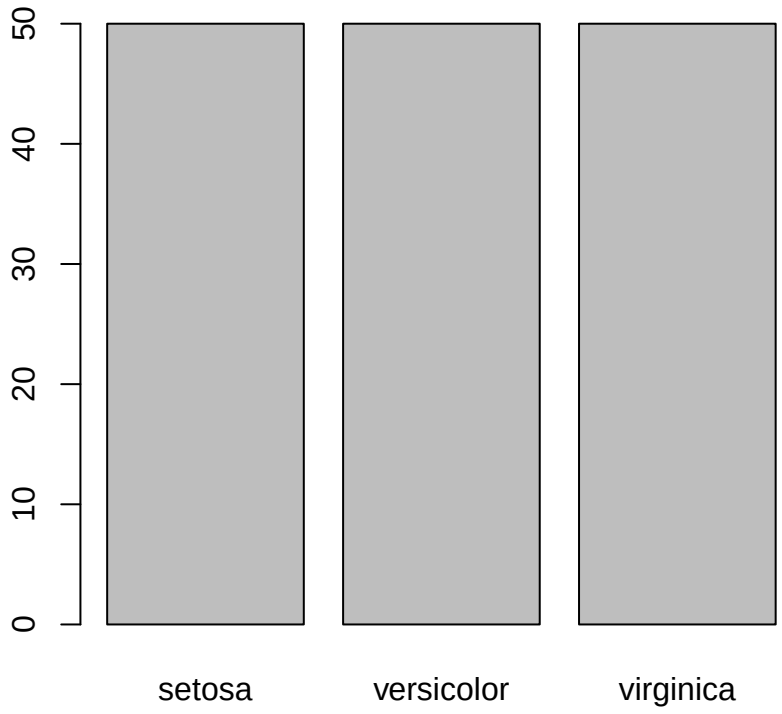
```
sltable=table(sl) # odd because so many different measurements
sltable
```

```
sl
4.3 4.4 4.5 4.6 4.7 4.8 4.9   5 5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9   6 6.1 6.2
  1   3   1   4   2   5   6  10   9   4   1   6   7   6   8   7   3   6   6   4
6.3 6.4 6.5 6.6 6.7 6.8 6.9   7 7.1 7.2 7.3 7.4 7.6 7.7 7.9
  9   7   5   2   8   3   4   1   1   3   1   1   1   4   1
```

Iris barplots of Species I

Default grey color, no title

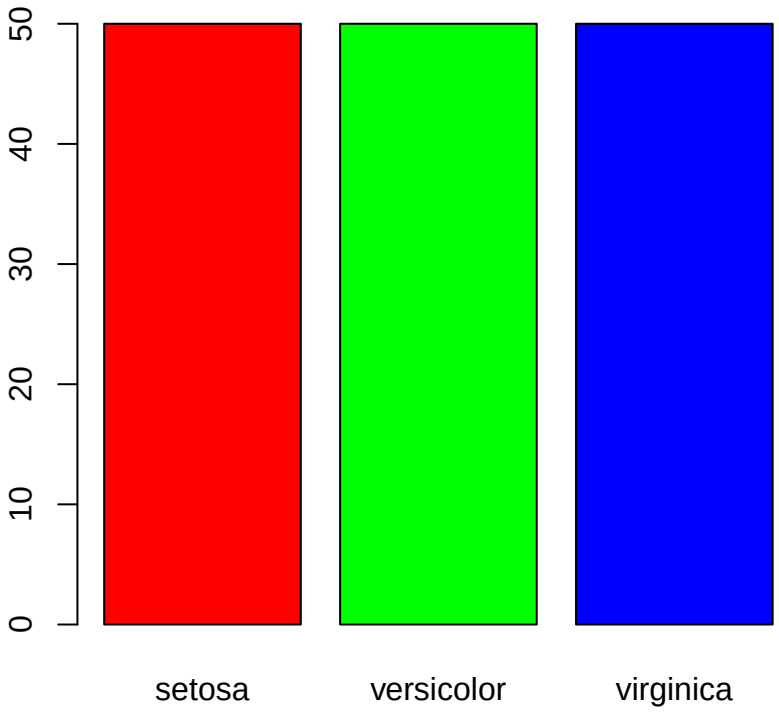
```
barplot(sptable)
```



Iris barplots of Species II

Colors!

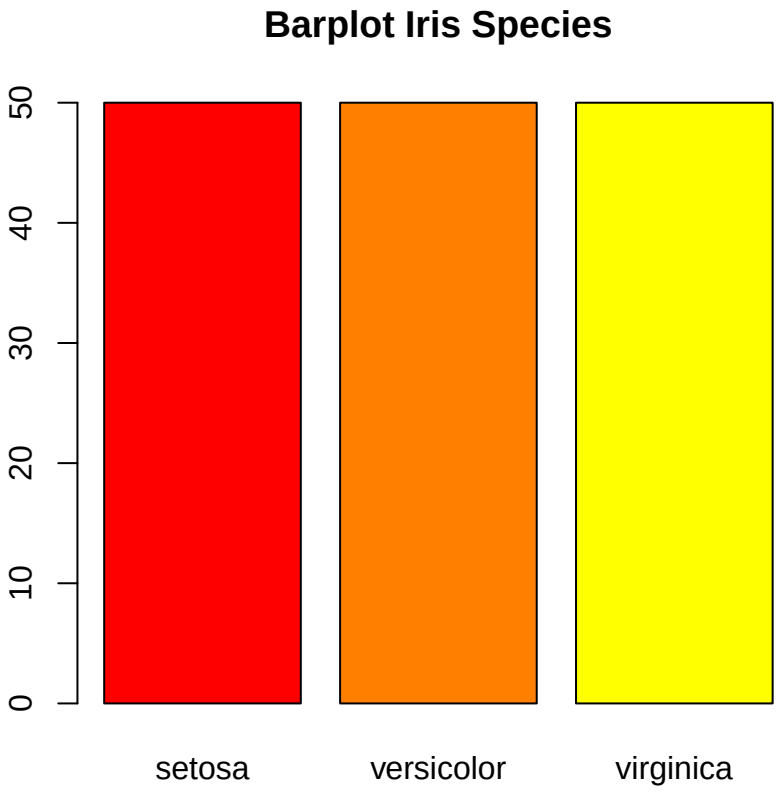
```
barplot(sptable,col=rainbow(3)) # the value here would be different if more species types
```



Iris barplots of Species III

Yay more colors and a title

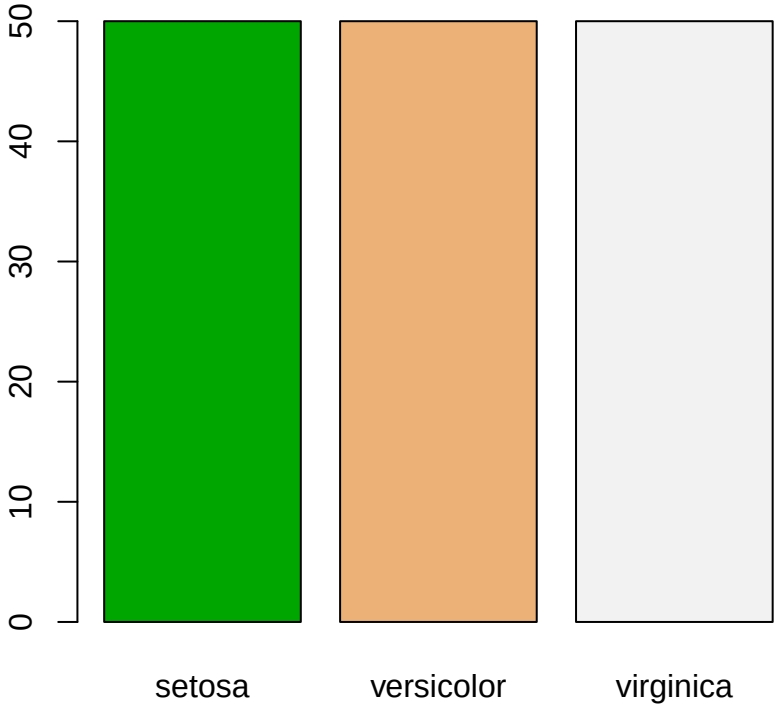
```
barplot(sptable,col=heat.colors(3),main='Barplot Iris Species')
```



Iris barplots of Species IV

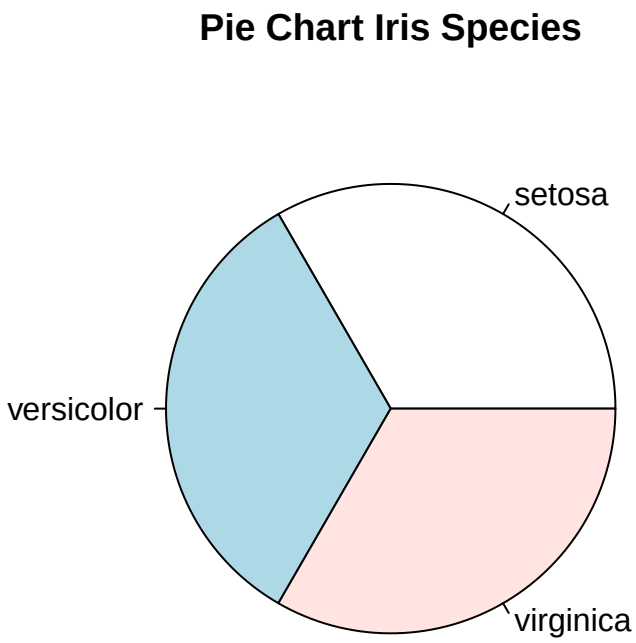
OK, last set of colors (there are more but I will stop the torture)

```
barplot(sptable,col=terrain.colors(3))
```



Iris pie charts of Species

```
pie(sptable,main='Pie Chart Iris Species') # default colors are ok
```



Basic Subsetting I

One thing we may have to do at times (ok. . . many times) is subset the dataset. There is an entire module for this later but here will be the first introduction to basic subsetting.

When you look at the `Species` variable, there are three different species listed: setosa, verisolor, and virginica. To see `sepal length` by `species`, we need some logical comparisons of the elements within a vector (and we want to keep data points aligned with their species).

Basic Subsetting II

To look at the `species` vector and only display the setosa species, we use:

```
species=='setosa'
```

The double equal sign (`==`) is a logical operator (logical meaning T/F). It compares every element of `species` to the text character `setosa` and produces a logical vector in response containing `TRUE` where there are `setosa`s and `FALSE` where there are other names.

```
species=='setosa'
```

```
  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [49]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[145] FALSE FALSE FALSE FALSE FALSE FALSE
```

Basic Subsetting III

Now we want to align specific species with measurements of sepal and petal lengths and widths. To do that we will access one vector by a specified condition of a different variable (sepal and petal lengths and widths by species).

```
sl[species=='setosa']
```

The square brackets `[]`are extracting only the sepal length measurements from the setosa species only. From here, we can also create another vector with the assignment statement.

Basic Subsetting IV

```
# all sepal lengths by species
setosa.sl=sl[species=='setosa']
virginica.sl=sl[species=='virginica']
versicolor.sl=sl[species=='versicolor']
# all sepal widths by species
setosa.sw=sw[species=='setosa']
virginica.sw=sw[species=='virginica']
versicolor.sw=sw[species=='versicolor']
```

Basic Subsetting V

```
# all petal lengths by species
setosa.pl=pl[species=='setosa']
virginica.pl=pl[species=='virginica']
versicolor.pl=pl[species=='versicolor']
# all petal widths by species
setosa.pw=pw[species=='setosa']
virginica.pw=pw[species=='virginica']
versicolor.pw=pw[species=='versicolor']
# look at one
versicolor.pl
```

```
 [1] 4.7 4.5 4.9 4.0 4.6 4.5 4.7 3.3 4.6 3.9 3.5 4.2 4.0 4.7 3.6 4.4 4.5 4.1 4.5
[20] 3.9 4.8 4.0 4.9 4.7 4.3 4.4 4.8 5.0 4.5 3.5 3.8 3.7 3.9 5.1 4.5 4.5 4.7 4.4
[39] 4.1 4.0 4.4 4.6 4.0 3.3 4.2 4.2 4.2 4.3 3.0 4.1
```

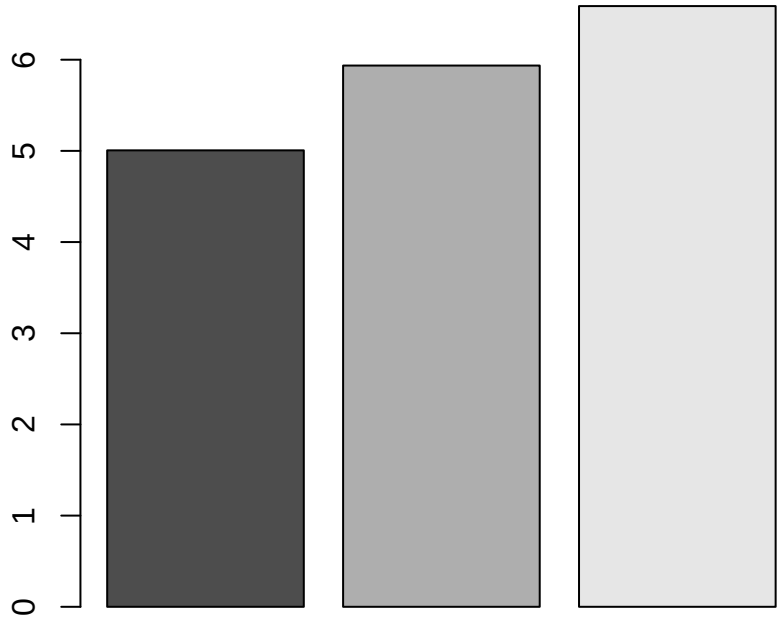
Basic Subsetting VI

All that work and we will finally create something. We will calculate the means of the numerical variables by species and finally make a barplot with that information.

`mean(x)` where x is a numeric vector. Remember that functions can be used within other functions.

Barplot of means (sepal lengths)

```
means.sl=c(mean(setosa.sl),mean(versicolor.sl),mean(virginica.sl))
barplot(means.sl,col=gray.colors(3))
```

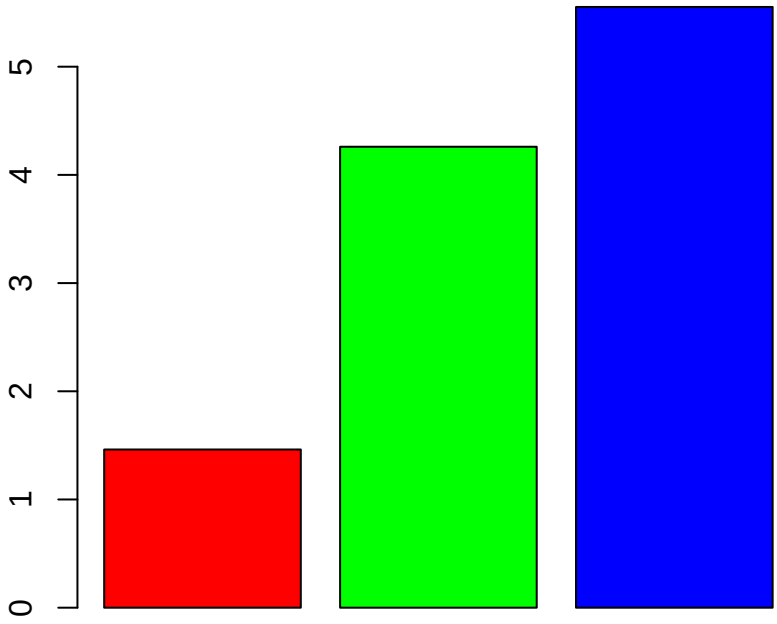


Barplot of means (sepal widths)

```
means.sw=c(mean(setosa.sw),mean(versicolor.sw),mean(virginica.sw))
barplot(means.sw,col=heat.colors(3))
```

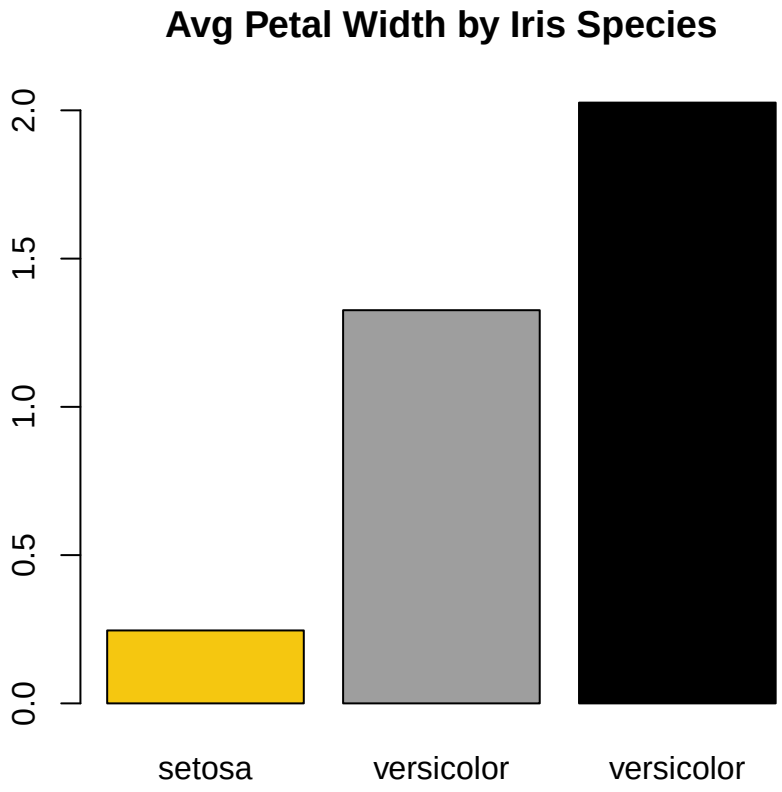
Barplot of means (petal lengths)

```
means.pl=c(mean(setosa.pl),mean(versicolor.pl),mean(virginica.pl))
barplot(means.pl,col=rainbow(3))
```



Barplot of means (petal widths)

```
means.pw=c(mean(setosa.pw),mean(versicolor.pw),mean(virginica.pw))
names=c('setosa','versicolor','versicolor')
barplot(means.pw,col=c(7,8,9),names.arg=names,
        main="Avg Petal Width by Iris Species")
```



One last pie chart

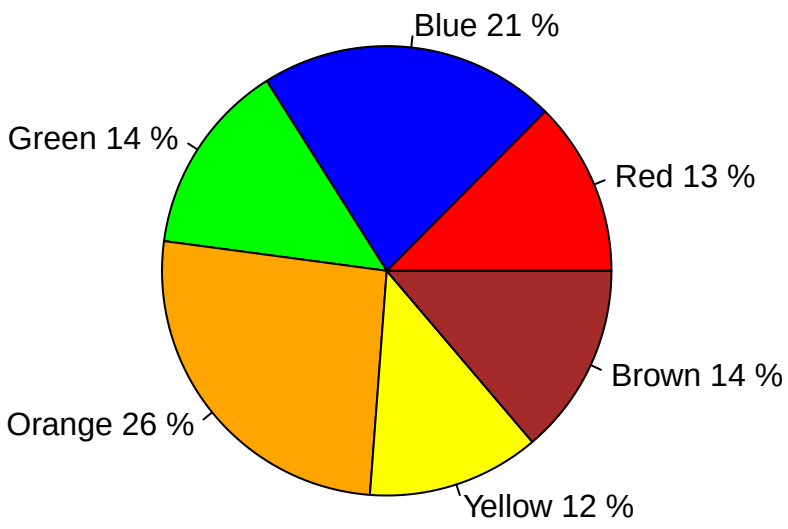
I hate these things (because they are easy to manipulate) but this one is M&Ms, so chocolate makes it ok.

```
colors=c('Red','Blue','Green','Orange','Yellow','Brown')
observed=c(92,157,102,190,91,101)
mars=c(.13,.24,.16,.2,.14,.13)
M.M=data.frame(colors,observed,mars)
pi=mars*sum(observed)
gcolors=c("red","blue","green",'orange','yellow','brown')
pct <- round(observed/sum(observed)*100)
lbls <- paste(colors,pct,"%",sep=" ") # add % to labels
```

Finally done with pie charts forever!

Seriously, do not use these things

```
pie(observed,labels = lbls, col=gcolors,
    main="Pie Chart of M&M colors")
```


Pie Chart of M&M colors
Blue 21 %
Red 13 %
Brown 14 %
Yellow 12 %
Orange 26 %
Green 14 %