Simple R

H Qin

simple calculations

```
10-7
```

```
## [1] 3
```

```
27/3.0
```

```
## [1] 9
```

```
35.46 *1.18
```

```
## [1] 41.8428
```

```
1+3
```

```
## [1] 4
```

The '#' sign inside the R code blocks indicates comments. Computer will ignore this line.

```
#3^2
3^2
```

```
## [1] 9
```

```
pi
```

```
## [1] 3.141593
```

```
pi^2
```

```
## [1] 9.869604
```

```
3.141593 * 3.141593
```

```
## [1] 9.869607
```

Calculator

```
#square
3^2
```

```
## [1] 9
```

```
5-2
```

```
## [1] 3
```

```
5/2
```

```
## [1] 2.5
```

```
5*3
```

```
## [1] 15
```

```
#Q: 5 to the 3rd power?
5^3
```

```
## [1] 125
```

```
# * means times
5*5*5
```

```
## [1] 125
```

```
#natural log, No 'ln'
log10(100)
```

```
## [1] 2
```

```
#exponetial function
exp(pi)
```

```
## [1] 23.14069
```

```
exp(0)
```

```
## [1] 1
```

```
#square-root
sqrt(pi)
```

```
## [1] 1.772454
```

```
sqrt(100)
```

```
## [1] 10
```

```
#Q, 100 to the 1/2 power
100^0.5
```

```
## [1] 10
```

Vectors or arrays

semicolon is not required, but is a good habit.

```
 #these are arrays (vectors)
#x = seq(0,10, 0.1)
#x;
x <- 5:9;  # = means assignment, x will stay in memory
x;
```

```
## [1] 5 6 7 8 9
```

```
1:15 #no assignment, no results stay in memory
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

```
z <- 1:15;
```

```
x = 3:10
x
```

```
## [1]  3  4  5  6  7  8  9 10
```

```
length(x)  #length() is an function in R
```

```
## [1] 8
```

```
#Q what does length() do?

help(length)
```

#look for helps

```
?seq
help(seq)
?length
```

= sign in programming

```
x = 3:10
x+1 #no assignment
```

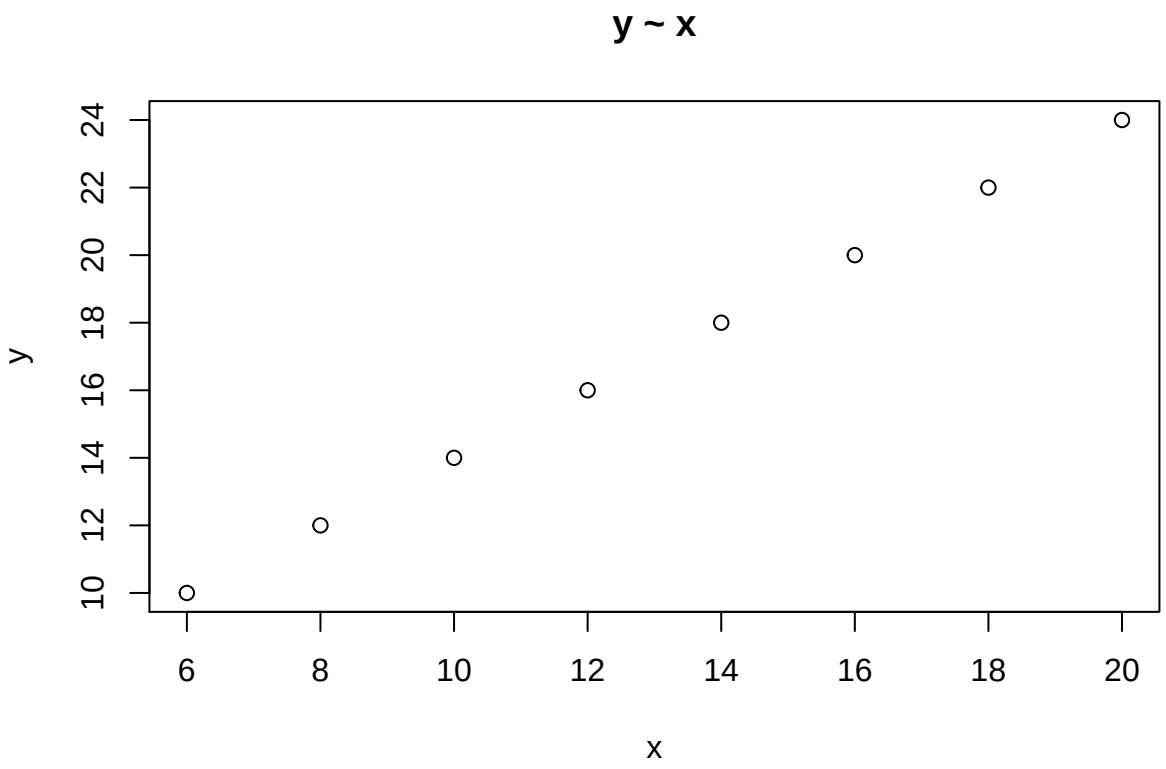
```
## [1]  4  5  6  7  8  9 10 11
```

```
x = x * 2; # what happens to x?
#The difference bw theese two lines is an important computing concept
# x =x+1, assign a new value from righthandside to the lefthandside.
x;
```

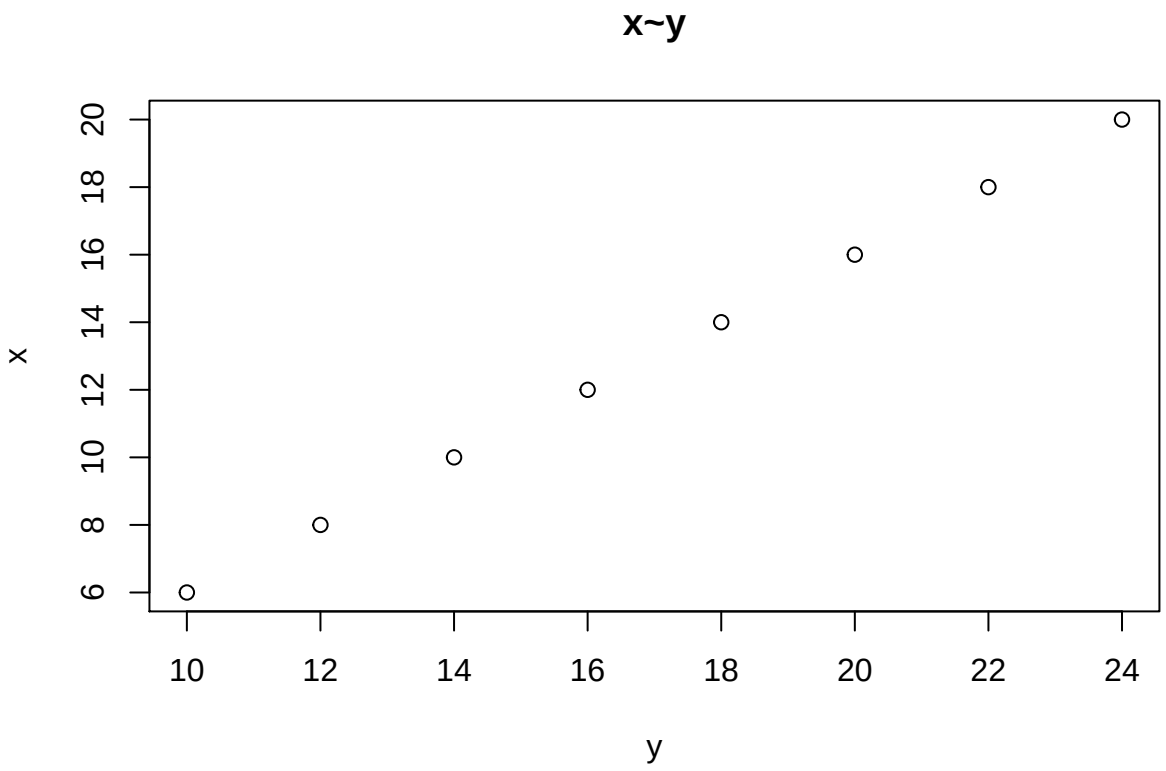
```
## [1]  6  8 10 12 14 16 18 20
```

ploting example

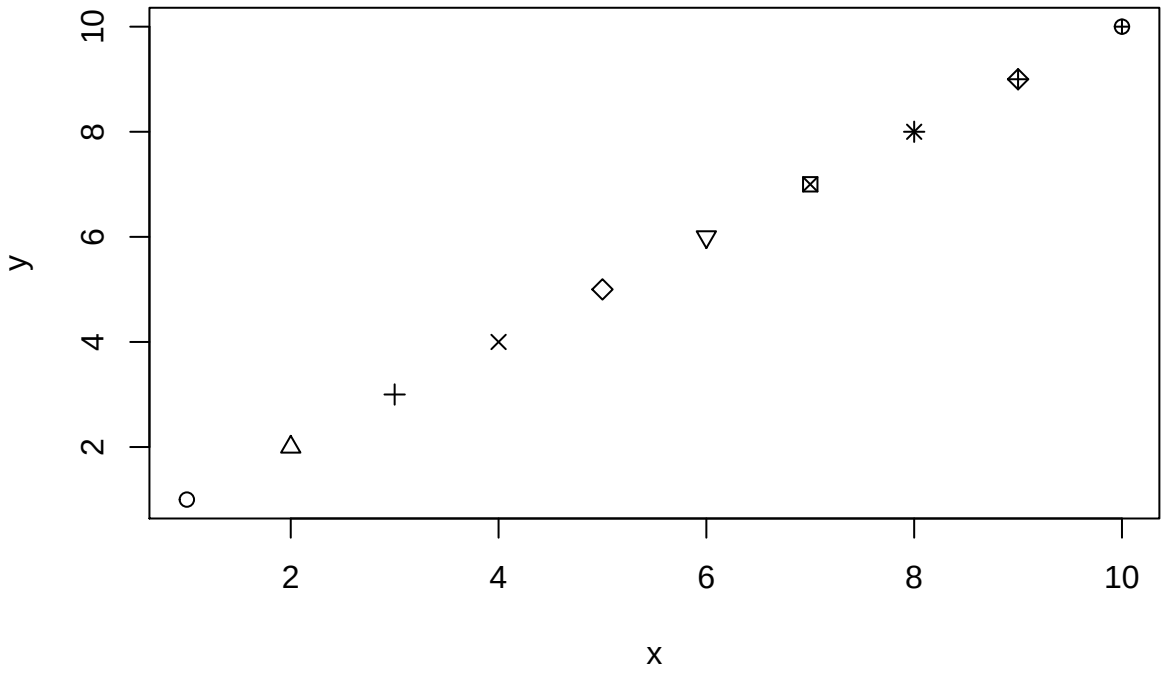
```
y = x+4
#simple plot
plot( y ~ x, main= "y ~ x" );
```




```
plot( x ~ y, main= "x~y" )
```




```
x = 1:10;
y = x;
plot( y ~ x, pch=x);
```



#If in R, try recording, page-down and up to see plot histories #Rstudio save plots by default

```
#plot( y ~ x, main="first plot" );
#plot( x ~ y, main="second plot" )
#?plot

#exercise
# modify plot( y ~ x ) to line plot
# by adding type into the command
# ... ...
#plot( y ~ x, main="line-point plot", type='b', pch=19 )


#this is another way of specifying an array
x = c( 0.1, 0.3, 1, 3, 5, 10, 0.001, 0.913 );
w = c(1, 3, 5, 7)
x[4:6]
```

```
## [1]  3  5 10
```

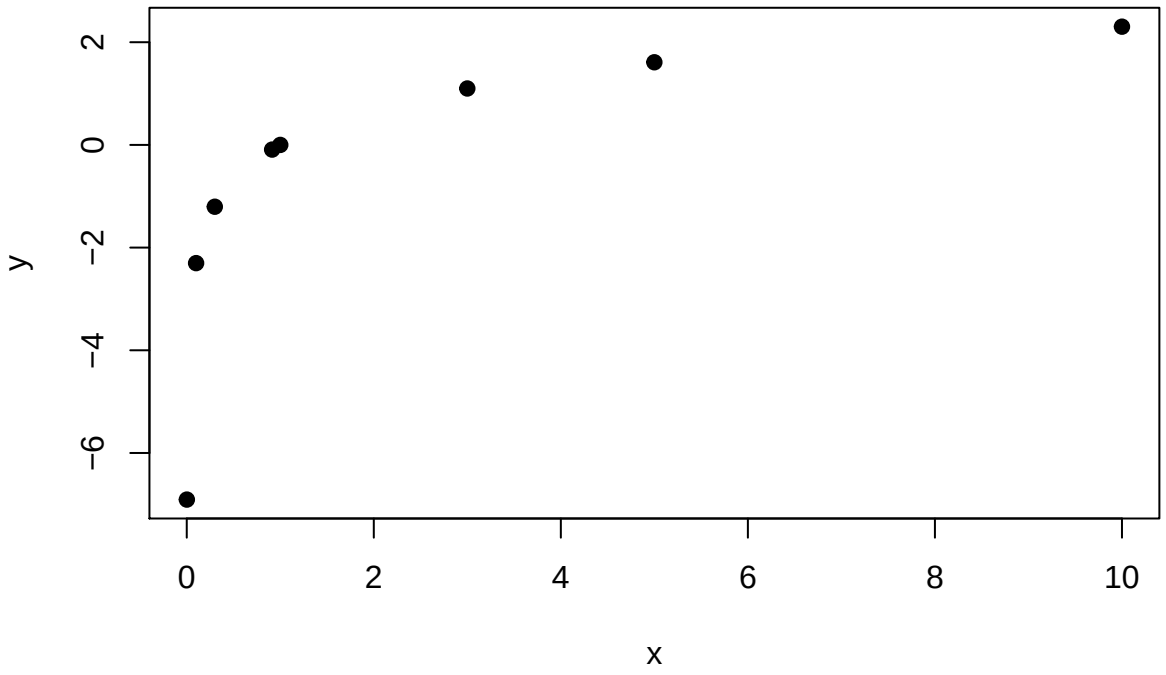
```
x[2]
```

```
## [1] 0.3
```

```
x[c(1,5,2)]
```

```
## [1] 0.1 5.0 0.3
```

```
y = log(x);
plot( y ~ x, pch=19 );
```




```
mycolors = c("red", "lightblue", "blue", "gold")
x = 1:4
y = x + 2
plot( y ~ x, col=mycolors, pch=19)
```