STAT2008

Do, While, If statements and other useful functions

Functions

`if(cond) {`*statements*`} else {`*statements*`}`	evaluate condition
`for(var in seq) {`*statements*`}`	execute one loop for each var in seq
`while(cond) {`*statements*`}`	execute loop as long as condition is true
`repeat {`*statements*`}`	execute expression on each loop
`break`	exits loop
`print(x)`	prints object x to screen
`stop("...")`	stop function and print error message
`warning("...")`	generate warning message

Grouping, loops and conditional execution

- Control statements
  - if statements

  - The language has available a conditional construction of the form

    if (expr 1) expr 2 else expr 3

where expr 1 must evaluate to a logical value

Repetitive execution

- for loops, repeat and while
  - for (name in expr 1) expr 2
  - where name is the loop variable.
  - expr 1 is a vector expression, (eg a sequence like 1:10)
  - expr 2 is repeatedly evaluated as name ranges through the values in the vector result of expr 1.

Repetitive execution

- Other looping facilities include the
  - repeat expr statement
  - while (condition) expr statement.
  - The break statement - used to terminate any loop
  - The next statement can be used to discontinue one particular cycle and skip to the “next”.

For

- When the same or similar tasks need to be performed multiple times

```
> for(i in 1:10) {
+    print(i*i)
+ }
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
[1] 49
[1] 64
[1] 81
[1] 100
```

For and If

```
for(i in 10:0) {
     if (i==5){
    print("Hoorah we are halfway
    through!!")
     }
    else
    {
    print(i)
    }
    }
```

```
[1] 10
[1] 9
[1] 8
[1] 7
[1] 6
[1] "Hoorah we are halfway
    through!!"
[1] 4
[1] 3
[1] 2
[1] 1
[1] 0
>
```

While

```
> i=99
> while(i<=103){
+ print(i)
+ i=i+1
+ }
[1] 99
[1] 100
[1] 101
[1] 102
[1] 103
```

rnorm

- Usage: rnorm(n, mean = 0, sd = 1)

Example:

```
> x<-rnorm(1,0,1)
> x
[1] -0.4642039
>
```

Repeat, Break, If

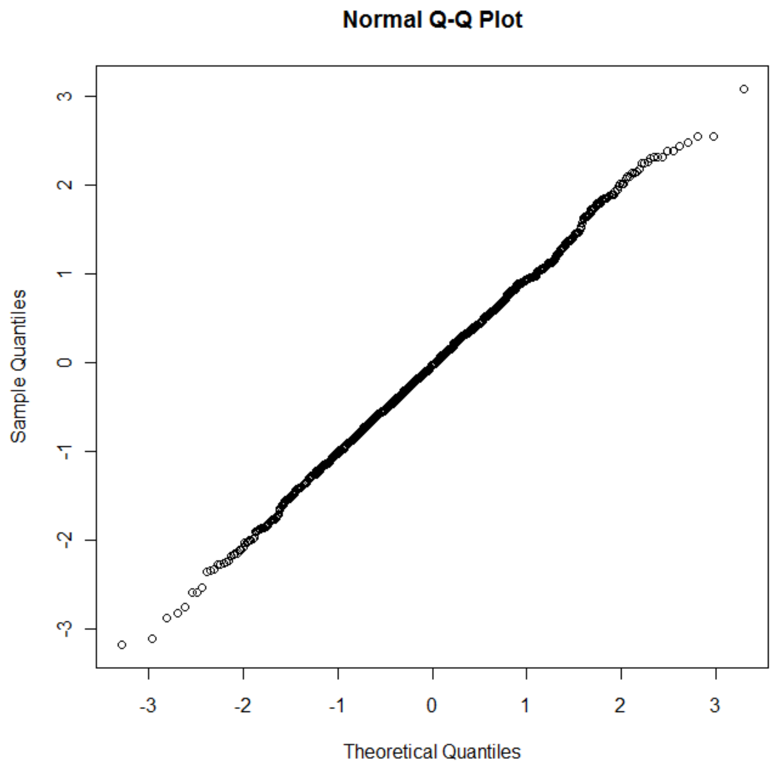
```
repeat {
x <- rnorm(1, 0, 1)
print(x)
if(x < 0){ break }
}
```

```
> repeat {
+ x <- rnorm(1, 0, 1)
+ print(x)
+ if(x < 0){ break }
+ }
[1] 0.2300458
[1] 1.253165
[1] -0.9555578
>
```

qqnorm

- qqnorm is a generic function the default method of which produces a normal QQ plot of the values.

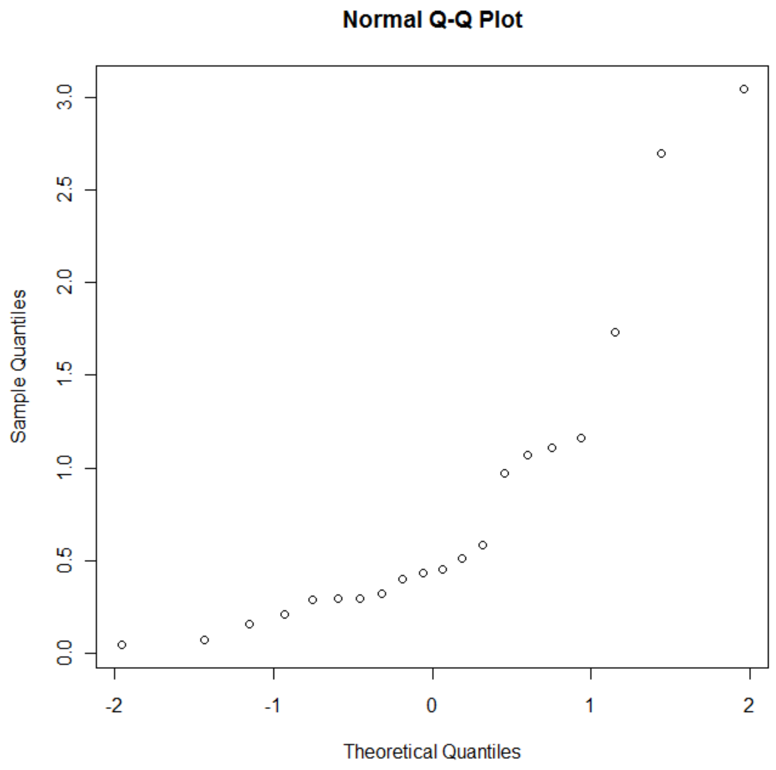
```
> x <- rnorm(1000, 0,
  1)
> qqnorm(x)
>
```



rexp

Density, distribution function, quantile function and random generation for the exponential distribution with rate where mean= 1/rate:

```
> ex <- rexp(20,1)
> qqnorm(ex)
>
```



User defined function betachng()

- Writing a function that investigates how the slope and intercept of a least-squares regression line change when several user-chosen data points are excluded, and we shall call it betachng().

- Such a function will be useful for identifying potential outliers and influential points in a data set.

betachng

```
> betachng <- function(resp,pred,excl){
+ exc <- unique(excl)
+ if(min(excl)<1) {
+ print("Invalid Point to be Excluded - Index too small") }
+ else if(max(excl)>length(pred)) {
+ print("Invalid Point to be Excluded - Index too large") }
+ else {
+ beta <- lsfit(pred,resp)$coef
+ beta.red <-lsfit(pred[-exc],resp[-exc])$coef
+ beta - beta.red }
+ }
```

Using the function

```
betachng(Height,Weight,1)
Intercept                     X
0.387648288      -0.007282526
```

What are these numbers?

Homework

- Using the help function in R, write a function that includes the stop and warning functions

Homework

- Using R help, investigate the difference between using:

- else if
- Vs
- else