Control Structures

- Blocks: Use braces {..} to group expressions together for function bodies and for...
- Conditional Execution:

```
if (n >= 25) {
  warning("n >=25, so using normal approximation")
  [ . . . code to use normal approximation . . . ]
} else {
  [ . . . special small sample code . . . ]
}
```

The logical operators && and || are useful in conditional expressions:

```
if (iters >= max.iters || accuracy < tolerance) {
  return(answer)
}
```

Also, there's a vector version:

```
> ifelse(c(T,T,F,T),c(1,2,3,4),c(10,20,30,40))
[1]  1  2 30  4
>
```


Control Structures

- Repetition:
  - `for` loop

```
> for (i in 1:1000) { [ . . . repeated 1000 times . . . ] }
> student.names <- c("Sally", "Joe", "William")
> for (name in c("Bill","Joe","Sally")) {
    if (!name %in% student.names)
      warning("Missing student: ", name)
  }
Warning message:
Missing student: Bill
>
```

  - `while` loop

```
while (accuracy < tolerance && iters < max.iters) {
  [ . . . repeat while condition holds . . . ]
}
```

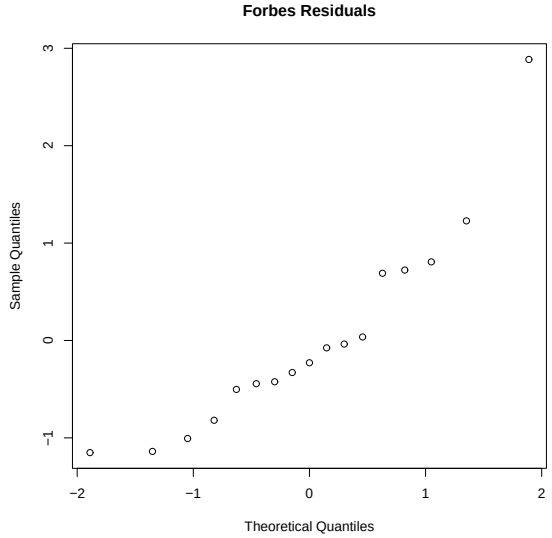
  - `repeat` loop

```
> repeat { print("I hate R!") }
```

Inside a loop, `next` moves on to the next iteration, `break` terminates the loop prematurely, and `return(...)` ends the loop and returns from the function.

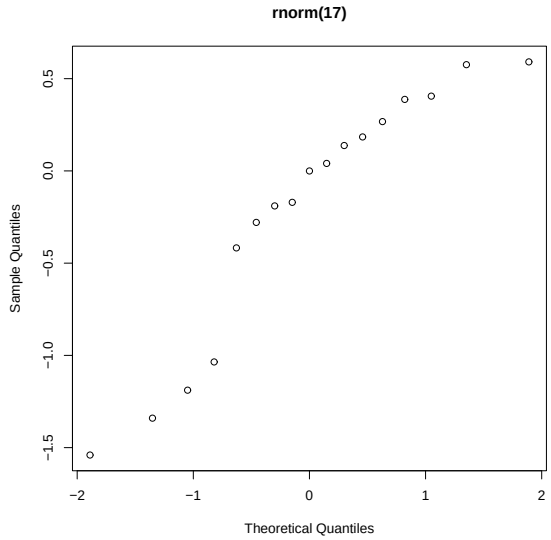
Evaluating a `qqnorm` Plot

```
> forbes.lm <- lm(pres ~ bp, data=forbes)
> r <- rstandard(forbes.lm)
> length(r)
[1] 17
> qqnorm(r, main="Forbes Residuals")
```



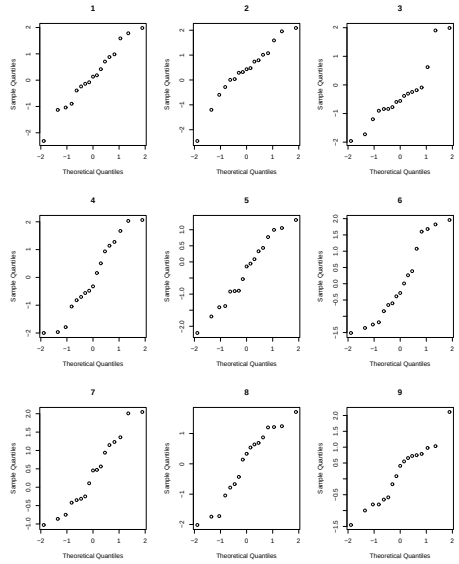
Evaluating a `qqnorm` Plot

```
> qqnorm(rnorm(17), main="rnorm(17)")
```



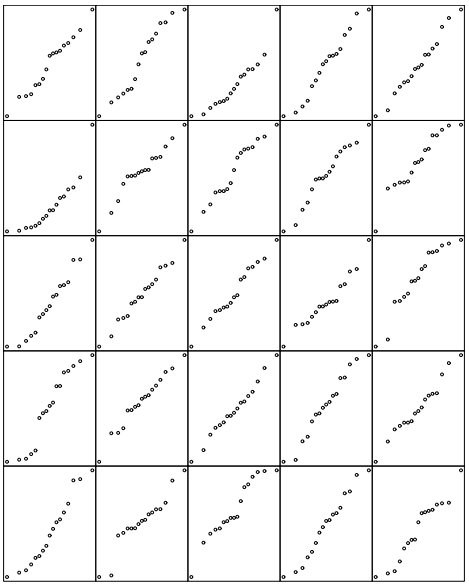
Evaluating a qqnorm Plot

```
> opar <- par(mfrow=c(3,3))
> for (i in 1:9)
    qqnorm(rnorm(17), main=i)
> par(opar)
```



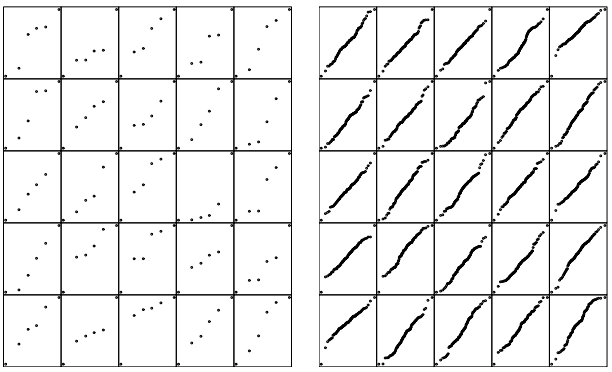
Evaluating a qqnorm Plot

```
> opar <- par(mfrow=c(5,5),mar=c(0,0,0,0),ann=F,xaxt="n",yaxt="n")
> for (i in 1:25)
    qqnorm(rnorm(17))
> par(opar)
```



Evaluating More qqnorm Plots

```
> qqnorm.sample <- function(n, nrow=5, ncol=5, annotate=F)
{
  if (annotate)
    opar <- par(mfrow=c(nrow,ncol),ask=F)
  else
    opar <- par(mfrow=c(nrow,ncol),ask=F,mar=c(0,0,0,0),ann=F,
                xaxt="n",yaxt="n")
  for (i in 1:(nrow*ncol)) {
    qqnorm(rnorm(n), main=i)
  }
  par(opar)
}
> qqnorm.sample(6)
> qqnorm.sample(100)
```



qqnorm Confidence Bounds

```
> print(qqnorm(sort(rnorm(5)),plot.it=F))
$x
[1] -1.1797611 -0.4972006  0.0000000  0.4972006  1.1797611
$y
[1] -1.7138151 -1.2574695 -0.7910166  0.3965363  0.8228089
> print(qqnorm(sort(rnorm(5)),plot.it=F))
$x
[1] -1.1797611 -0.4972006  0.0000000  0.4972006  1.1797611
$y
[1] -0.83008885 -0.04778618  0.14088821  0.20620674  0.41256715
> qqnorm.conf <- function(sample, simulations=100, ...)
{
  n <- length(sample)
  q <- qqnorm(sort(rnorm(n)), plot.it=F)
  x <- q$x
  y <- q$y
  for (i in 1:(simulations-1)) {
    q <-  qqnorm(sort(rnorm(n)),plot.it=F)
    y <- rbind(y, q$y)
  }
  bands <- apply(y, 2, quantile, probs=c(.025,.975))
  qqnorm(sample, ...)
  lines(c(x,NA,x), c(bands[1,],NA,bands[2,]), lty="dashed")
}
>
```

`qqnorm` Confidence Bounds

```
> qqnorm.conf(r, main="Forbes Standard Residuals")
> qqnorm.conf(r, main="Forbes Standard Residuals")
>
```