Az R adatelemzési nyelv alapjai II.

Egészségügyi informatika és biostatisztika

Gézsi András
gezsi@mit.bme.hu

Functions

Functions

- Functions do things with data
  - “Input”: function arguments (0,1,2,…)
  - “Output”: function result (exactly one)

- Why do we write functions?
  - To simplify code
  - To reduce code redundancy

- Example:

```
add <- function(a,b)
{ result <- a+b
  return(result) }
```

General Form of Functions

```
function(arguments)
{
      # do things…
      expression
}
```

```
function(arguments)
{
      # do things…
      return( expression )
}
```

Calling Conventions for Functions

- Arguments may be specified in the same order in which they occur in function definition, in which case the values are supplied in order.
- Arguments may be specified as name=value, when the order in which the arguments appear is irrelevant.
- Above two rules can be mixed.

```
> t.test(x1, y1, var.equal=F,
  conf.level=.99)
> t.test(var.equal=F, conf.level=.99,
  x1, y1)
```

Missing Arguments

- R function can handle missing arguments two ways:
    - either by providing a default expression in the argument list of definition, or
    - by testing explicitly for missing arguments.

Missing Arguments in Functions

```
> add <- function(x,y=0){x + y}
> add(4)

> add <- function(x,y){
  if(missing(y)) x
  else x+y
}
> add(4)
```

Variable Number of Arguments

- The special argument name “…” in the function definition will match any number of arguments in the call.

Variable Number of Arguments

```
> mean.of.all <- function(…){ mean(c(…)) }
> mean.of.all(1:10,20:100,12:14)

> mean.of.means <- function(…)
{
  means <- numeric()
  for(x in list(…))
      means <- c(means,mean(x))
  mean(means)
}
> mean.of.means(1:10,20:100,12:14)
```

Formulas

Formulas

- Formulation of models
- Expression: `y ~ model`
  - the response `y` is modelled by a linear predictor specified symbolically by `model`
  - operators
  - `+` => `term1+term2` (main effects)
  - `:` => `term1:term2` (interaction)
  - `*` => `term1*term2` = `term1+term2+term1:term2` (main effects and interaction)
- `?formula`

Graphics in R

plot()

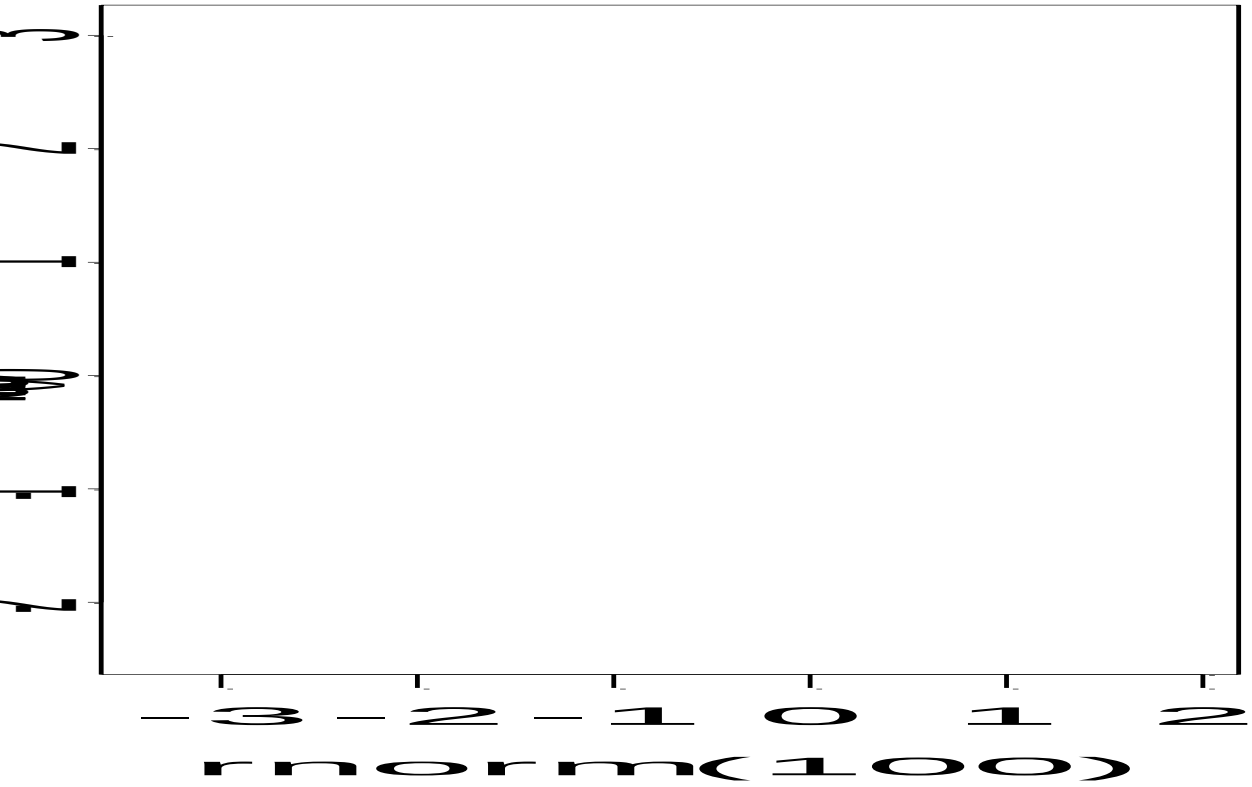
- If x and y are vectors, `plot(x,y)` produces a scatterplot of x against y.
- `plot(x)` produces a time series plot if x is a numeric vector or time series object.
- `plot(mtcars)`
  `plot(~ expr)`
  `plot(y ~ expr)`, where df is a data frame, y is any object, expr is a list of object names separated by `+' (e.g. a + b + c).
- The first two forms produce distributional plots of the variables in a data frame (first form) or of a number of named objects (second form). The third form plots y against every object named in expr.

plot()

```
> attach( mtcars )
> plot( mtcars )
> plot( ~ cyl +  hp + drat + wt )
> plot( mtcars,  mpg ~ cyl +  hp +
drat + wt )
```

Graphics with **`plot()`**

```
> plot(rnorm(100),rnorm(100))
```



The function **`rnorm()`** Simulates a random normal distribution .

Help **`?rnorm`**,
and **`?runif`**,
`?rexp`,
`?binom`, ...

Graphics with `plot()`

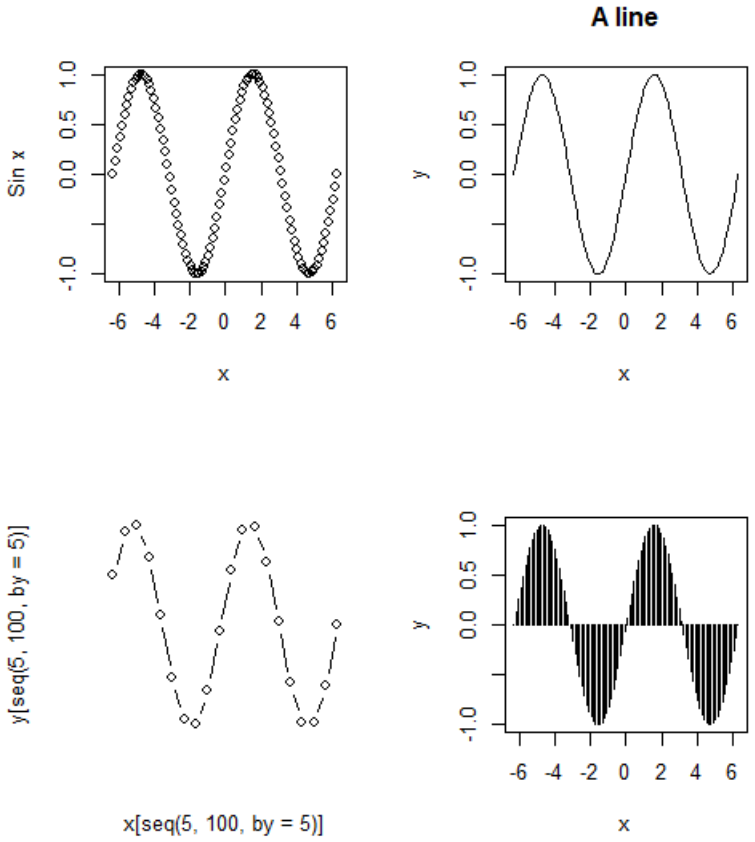
```
> x <- seq(-2*pi,2*pi,length=100)
> y <- sin(x)

> par(mfrow=c(2,2))
> plot(x,y,xlab="x",
    ylab="Sin x")

> plot(x,y,type= "l",
    main="A Line")

> plot(x[seq(5,100,by=5)],
  y[seq(5,100,by=5)],
  type= "b",axes=F)

> plot(x,y,type="h" )
> par(mfrow=c(1,1))
```



Graphical Parameters of `plot()`

```
type = "c": p (default), l,b,s,o,h,n
pch= "+": character or numbers 1 - 18
lty=1 : numbers #line type
lwd=2 : numbers #line width
xlab = "string", ylab = "string"
sub = "string", main = "string"
xlim = c(lo,hi), ylim = c(lo,hi)
```

And some more.

Graphical Parameters of `plot()`

```
x <- 1:10
y <- 2*x + rnorm(10,0,1)
plot(x,y,type="p") #Try l,b,s,o,h,n
     # axes=T, F
     # xlab="age", ylab="weight"
     # sub="sub title",
     # main="main title"
     # xlim=c(0,12), ylim=c(-1,12)
```

Other graphical functions

See also:

```
hist()
boxplot()

barplot()
image()
pairs()
persp()
piechart()
polygon()
```

Other graphical functions

```
> axis(1,at=c(2,4,5),
      legend("A","B","C"))
```

Axis details (“ticks”, legend, …)
Use **`xaxt="n"`** ou **`yaxt="n"`** inside **`plot()`**

```
> lines(x,y,…)
```

Line plots

```
> abline(lsfit(x,y))
> abline(0,1)
```

Add an adjustment
add a line of slope 1 and intercept 0

```
> legend(…)
> text(x,y,"Hey")
```

Legends: very flexible
Write text at coordinate x,y.

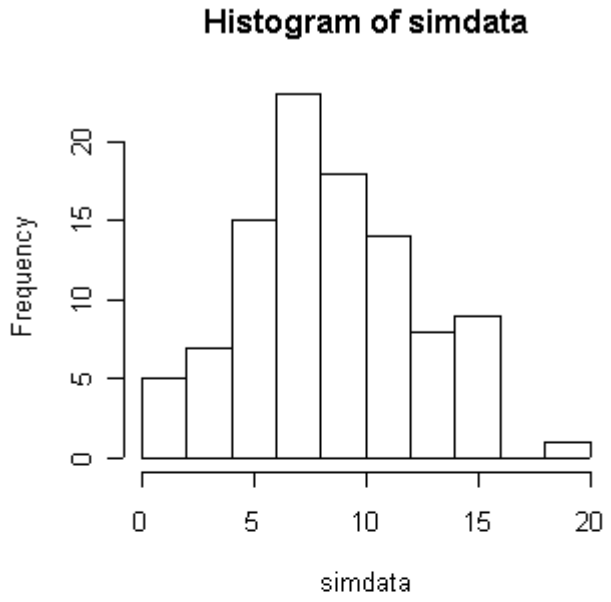
Histogram

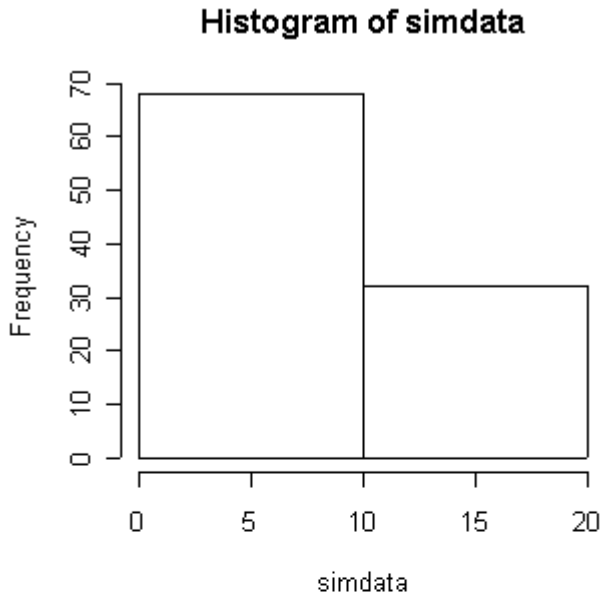
- A *histogram* is a special kind of bar plot
- It allows you to visualize the *distribution* of values for a numerical variable
- When drawn with a *density scale:*
  - the *AREA* (NOT height) of each bar is the proportion of observations in the interval
  - the *TOTAL AREA* is 100% (or 1)

Making a histogram

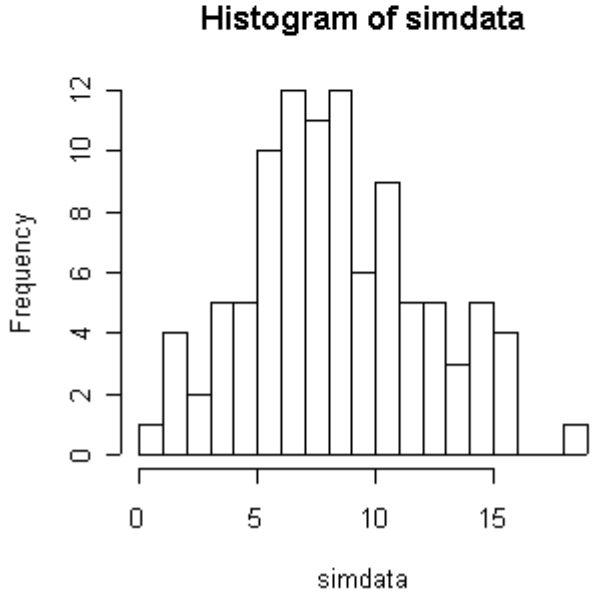
- Type **?hist** to view the help file
  - Note some important arguments, esp **breaks**
- Simulate some data, make histograms varying the number of bars (also called 'bins' or 'cells'), e.g.

```
> par(mfrow=c(2,2))    # set up multiple
  plots
> simdata <- rchisq(100,8)
> hist(simdata)  # default number of bins
> hist(simdata,breaks=2)  # etc,4,20
```


Histogram of simdata
Frequency
0
5
10
15
20
0
5
10
15
20
simdata

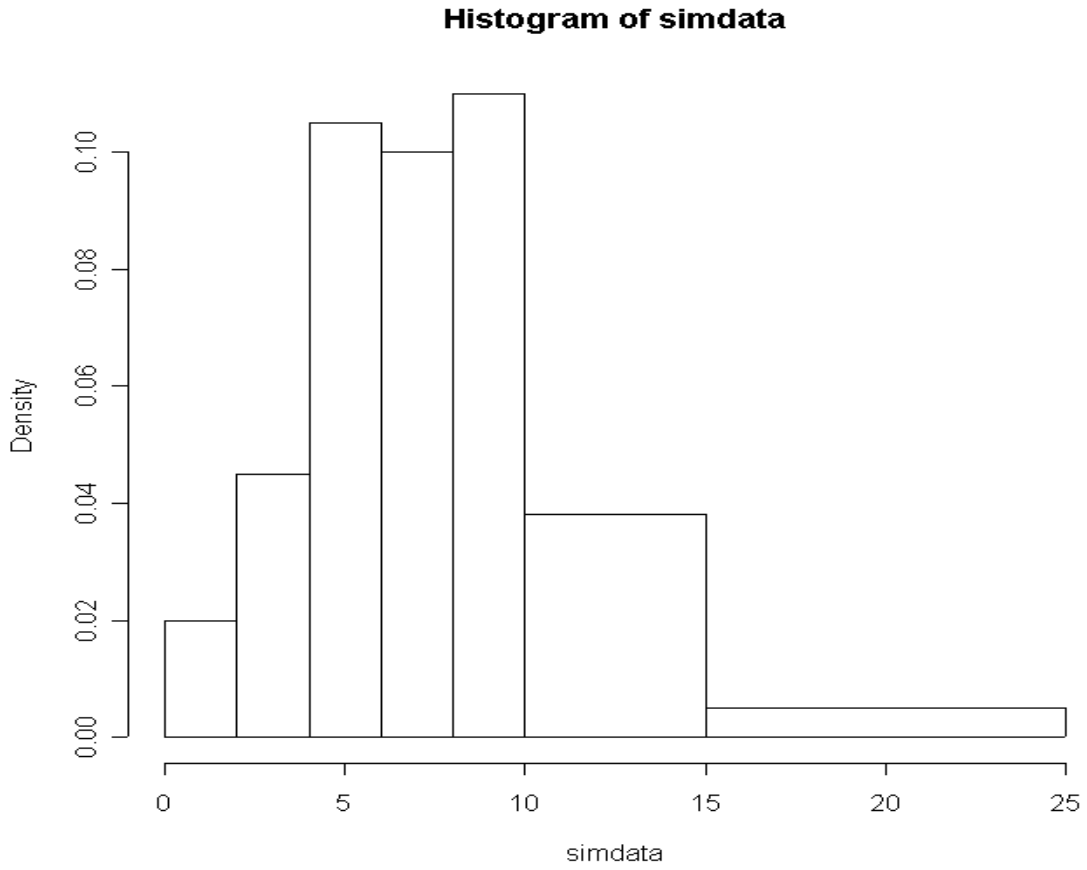

Histogram of simdata
Frequency
0
10
20
30
40
50
60
70
0
5
10
15
20
simdata


Histogram of simdata
Frequency
0
10
20
30
40
50
0
5
10
15
20
simdata


Histogram of simdata
Frequency
0
2
4
6
8
10
12
0
5
10
15
simdata

Setting your own breakpoints

```
> bps <- c(0,2,4,6,8,10,15,25)
> hist(simdata,breaks=bps)
```



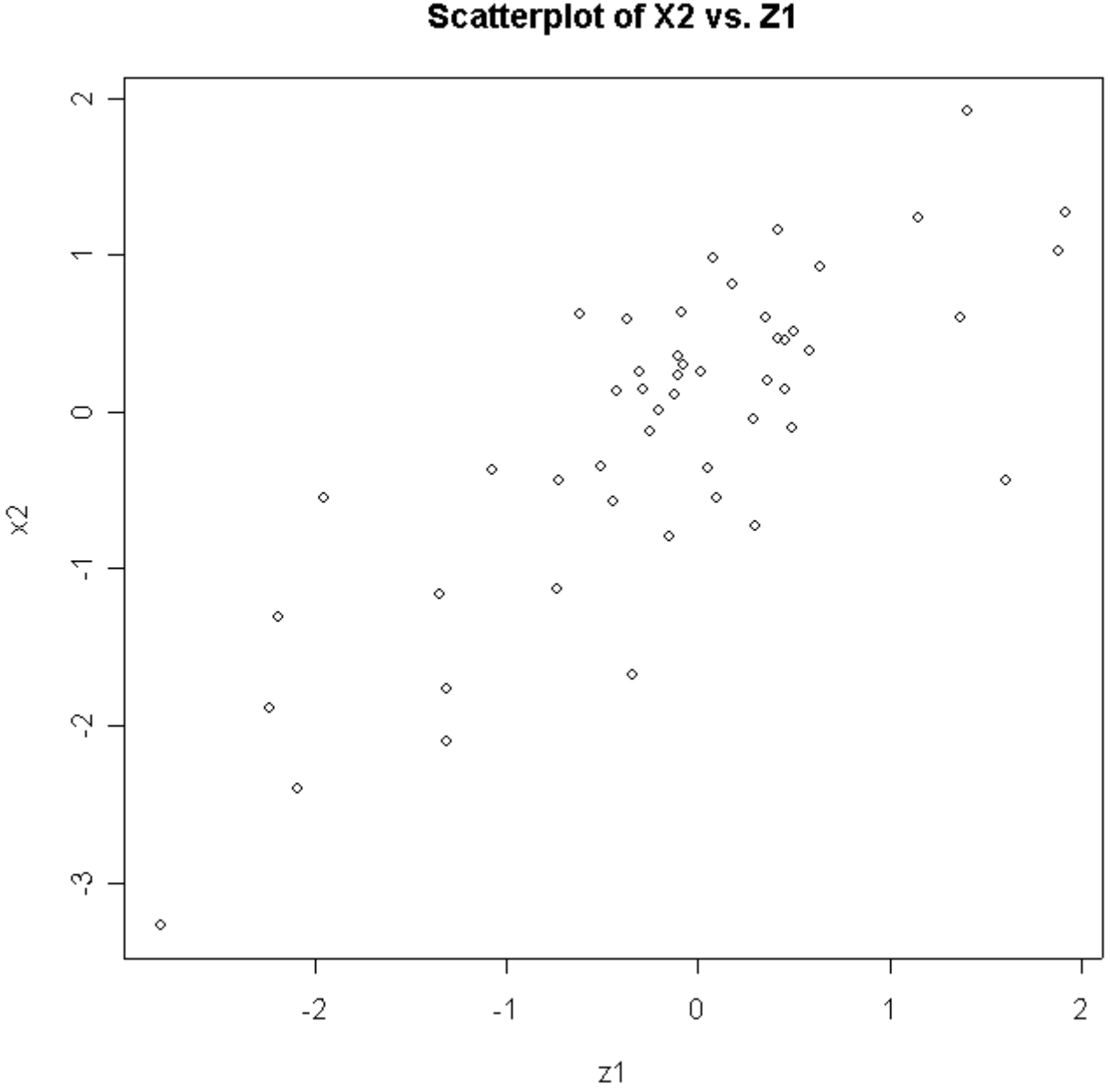
Scatterplot

- A scatterplot is a standard two-dimensional (X,Y) plot
- Used to examine the relationship between two (continuous) variables
- It is often useful to plot values for a single variable against the order or time the values were obtained

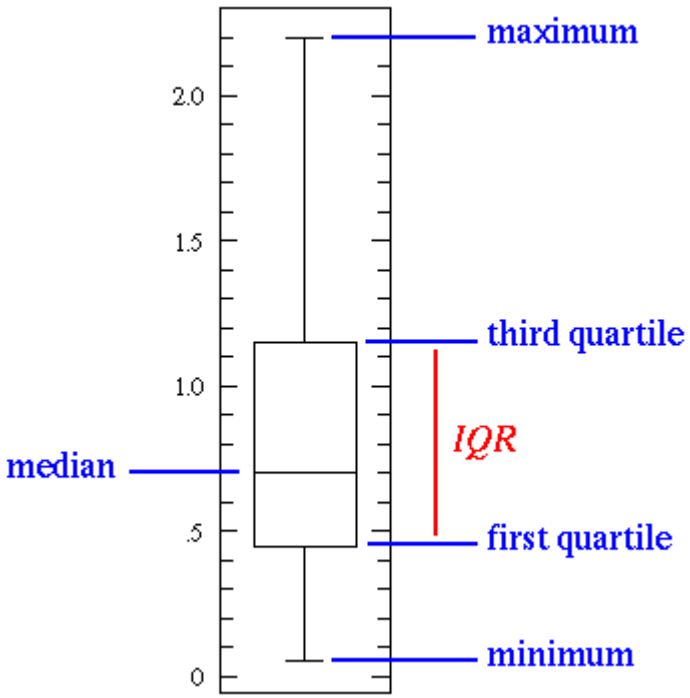
Making a scatterplot

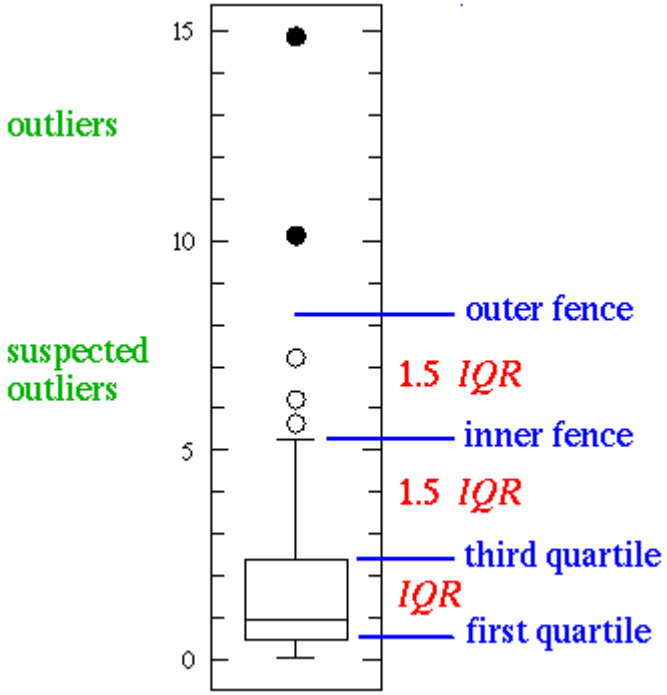
- Type `?plot` to view the help file
- Simulate a bivariate data set:

```
> z1 <- rnorm(50)
> z2 <- rnorm(50)
> rho <- .75          # (or any number between -1 and 1)
> x2 <- rho*z1+sqrt(1-rho^2)*z2
> plot(z1,x2)
```

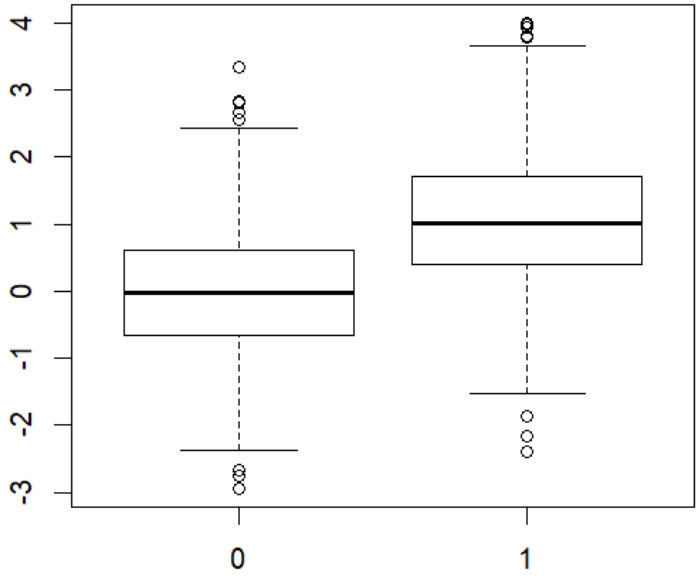

Scatterplot of X2 vs. Z1
x2
z1
2
1
0
-1
-2
-3
-2
-1
0
1
2

Boxplot (a.k.a. box and whisker diagram)





Making a boxplot



- Type **?boxplot** to view the help file
  - Split data by using formulas

- Simulate some data

```
> X <- c(rep(0,1000),rep(1,1000))
> Y <- X + rnorm(2000)
> simdata <- data.frame( X=X,Y=Y )
> boxplot( formula = Y ~ factor(X),
           data = simdata )
```